黑马程序员™ www.itheima.com - 黑马程序员

# 软件测试面试宝典

Version 2.0

**欢迎添加黑马华伟老师微信领取更多干货资料**

**黑马测试自助互动群也在等你哦**

扫一扫上面的二维码图案，加我微信

# 第一章　内容介绍

# 第二章　测试理论

## 一、 软件工程

### 1. 阐述软件生命周期都有哪些阶段？常见的软件生命周期模型有哪些？

软件生命周期是指一个计算机软件从功能确定、设计，到开发成功投入使用，并在使用中不断地修改、增补和完善，直到停止该软件的使用的全过程（从酝酿到废弃的过程）

生命周期从收到应用软件开始算起，到该软件不再使用为止。它有如下各方面的内容： 初始构思、需求分析、功能设计、内部设计、文档计划、测试计划、文档准备、集成、测 试、维护、升级、再测试、逐步淘汰 (phase-out)、等等

瀑布模型，迭代式模型，快速原型模型，螺旋模型

### 2. 什么是版本控制，常用的版本控制系统有哪些？

版本控制（Revision control）是一种软体工程技巧，籍以在开发的过程中，确保由不同人所编辑的同一档案都得到更新。

Git(读音为/gɪt/。)是一个开源的分布式版本控制系统，可以有效、高速的处理从很小到非常大的项目版本管理。Git 是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件。https://git-scm.com/doc

SVN 是 Subversion 的简称，是一个开放源代码的版本控制系统，相较于 RCS、CVS，它采用了分支管理系统，它的设计目标就是取代 CVS。互联网上很多版本控制服务已从 CVS 迁移到 Subversion。https://tortoisesvn.net/support.html

### 3. 简述软件测试与软件开发之间的关系？

（1）项目规划阶段：负责从单元测试到系统测试的整个测试阶段的监控。

（2）需求分析阶段：确定测试需求分析、系统测试计划的制定，评审后成为管理项目。测试需求分析是对产品生命周期中测试所需求的资源、配置、每阶段评判通过的规约；系统测试计划则是依据软件的需求规格说明书，制定测试计划和设计相应的测试用例。

（3）详细设计和概要设计阶段：确保集成测试计划和单元测试计划完成。

（4）编码阶段：由开发人员进行自己负责部分的代码的测试。在项目较大时，由专人进行编码阶段的测试任务。

（5）测试阶段（单元、集成、系统测试）：依据测试代码进行测试，并提交相应的测试状态报告和测试结束报告。

开发和测试是一个有机的整体！在产品的发布之前，开发和测试是循环进行的， 测出的缺陷要经开发人员修改后继续测试。在开发的同时测试经理开始编写测试用例，测 试文档要参考开发文档，所以开发和测试是不可分割的，

少了任何一个都不能开发出产品。

从角色方面看，像理论和实验的关系，开发人员通过自己的想象创造出一套思想，之 后测试人员再对它进行检验、证伪，开发人员再修改的过程从而不断丰富产品。从方法方 面看，是演绎和归纳的关系，一个要掌握大量的技术，一个要不断的从实例中学习。因这 两方面的不同，所以开发和测试看上去做的工作很不一样。

开发与测试是相辅相承、密不可分的，开发人员开发出新的产品后要通过测试判断产 品是否完全满足用户的需求。如果发现缺陷，提交给开发人员进行修复，然后再转交测试 人员进行回归测试，直到产品符合需求规格说明。一个符合用户需求的产品是开发和测试 共同努力的成果。

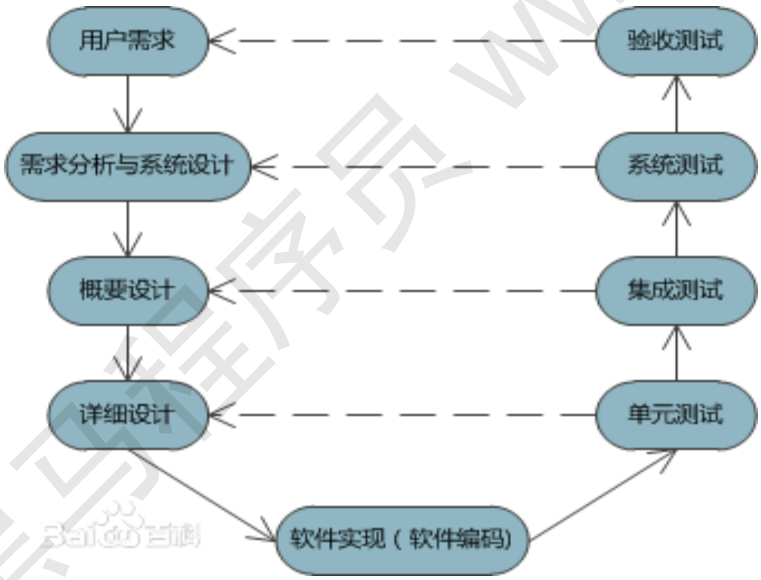
### 4. 什么是软件测试，定义和目的？

目的：用最少的人力、物力、时间来找到软件的错误并修复，从而降低商业风险

定义：使用人工和自动手段来检测软件是否满足需求

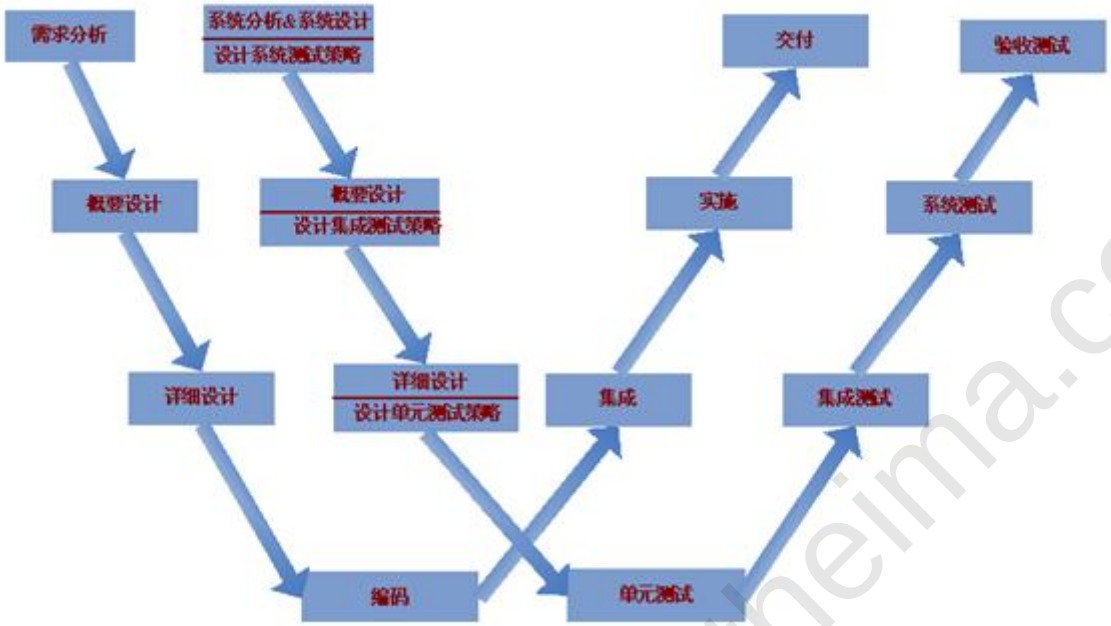
## 二、 测试模型

### 5. 常见测试模型有哪些？




特点：这是一种古老的瀑布模型，反映了实际和测试之间的关系

局限：仅仅把测试过程作为编码之后的一个阶段，忽视了测试对需求分析,系统设计的验证，如果前面设计错误，得一直到后期的验收测试才被发现，耗时耗力。




特点：测试与开发同时进行，在 V 模型的基础上，增加了在开发阶段的同步测试

局限：仍然不支持迭代，减少了一定错误发生率，但是需按照流水线进行设计、编码和测试

## 6. 请根据”V”模型分别概述测试人员在软件的需求定义阶段、设计阶段、编码阶段、系统集成阶段的工作任务及其相应生成的文档？

统集成阶段的工作任务及其相应生成的文档？

需求定义阶段：根据项目需求提取测试需求 并形成测试需求文档，根据提取的测试需求和项目计划进行测试计划的拟定，测试计划文档

设计阶段：根据测试需求拟定测试方案并形成测试方案文档；根据测试方案制定测试用例，并形成测试用例文档

编码阶段：执行测试并完善测试用例文档

系统集成阶段：测试总结报告，阶段问题统计报告，测试问题报告

## 7. W 模型的描述？

软件测试主要内容是对软件正确性、完整性、安全性和质量确认及验证。为了验证这些，软件测试是与开发同步进行的，它们之间的关系同步进行一一对应，当开发进行需求分析、概要设计、详细设计、编码实现、模块集成、系统构建与实施、交付运行时，测试人员对应工作是需求测试、概要设计测试、详细设计测试、单元测试、集成测试、系统测试、验收测试。可以通过 W 模型[3]展示如图 2.3 所示：

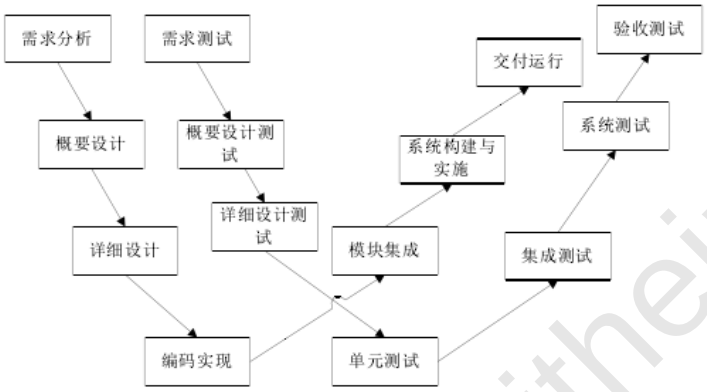



图 2.3 W 模型

图中显示 W 模型增加了软件开发各阶段中同步进行的验证和确认活动，测试的活动与软件开发整体是同步进行，测试的对象不仅仅是程序，还包括需求和设计，有利于尽早地全面的发现问题可降低软件开发和成本。

# 三、 测试计划

## 8. 编写测试计划的目的是？

使测试工作顺利进行；使项目参与人员沟通更舒畅；使测试工作更加系统化。

## 9. 测试计划编写的六要素？

why——为什么要进行这些测试

what—测试哪些方面，不同阶段的工作内容

when—测试不同阶段的起止时间

where—相应文档，缺陷的存放位置，测试环境等

who—项目有关人员组成，安排哪些测试人员进行测试

how—如何去做，使用哪些测试工具以及测试方法进行测试。

## 10. 项目版本执行过程中，测试人员如何把控测试进度？

在项目的系统测试过程中，测试负责人要及时了解测试进度，跟踪 BUG 提交、修复及验证情况以及系统的拷机

情况。

在开发初期阶段，测试组执行 BBFV 时，很多模块、功能点的开发完成进度和原计划会存在一定的偏差，就需要测试负责人动态的刷新 WBS 计划，根据实际的开发进度调整测试计划。

在开发阶段，存在版本编译不出来导致无法测试，开发人员修复代码太随意导致版本稳定性反复，需求变更过大导致后端测试开发变更严重等现象，会导致测试工作无法正常进行。就需要测试负责人及时反馈出来，根据项目本身的特点进行对应的处理。

当测试进度出现延期时，要及时确认问题原因，如果是问题协查导致，则需及时与研发人员进行沟通协商，看问题是否必须在测试环境进行排查，若为必现问题可与研发协商要求其在自己环境进行排查，若必须占用测试环境，则需及时调整测试计划，若因此可能影响版本的发布，则应及时与 SE 确认。

若发现有较多 BUG 未解决，则应主动联系 SE 及研发人员召开 BUG 会确定问题的解决时间。若发现有较多 BUG 未验证，则应提醒项目组的测试人员及时进行验证，对于一些拷机或非必现的 BUG，建议测试人员在此 BUG 上现做拷机标记，连续拷机一周未再复现的做关闭处理，若再次复现则继续进行排查。

疑难问题的跟控：比较难复现的问题，怎么去尝试复现。比较难定位的问题，怎么驱动、反馈给 SE，协调开发人员定位问题。比较难处理的问题，怎么跟控反馈进度等

每天下班前需确认拷机内容，每天上班第一件事需确认拷机结果，只有这样才能保证拷机的效果，实现拷机的真正意义。

### 11. 测试人员在软件开发过程中的任务是什么？

寻找 Bug；避免软件开发过程中的缺陷；衡量软件的品质；关注用户的需求。总的目标是：确保软件的质量。

## 四、 测试类型

### 12. 请列出你所知道的软件测试种类，至少 5 项？


## 13. 黑盒测试、白盒测试、单元测试、集成测试、系统测试、验收测试的区别与联系？

黑盒测试：把测试对象当成一个黑盒子，测试人员完全不考虑逻辑结构和内部特性， 只依据程式的需求说明书来检查程序的功能是否满足它的功能说明。

白盒测试：把测试对象当成一个透明的盒子，允许测试人员利用程序内部逻辑结构及 相关信息，设计或选择测试用例，对程序所有逻辑路径进行测试。

单元测试：白盒测试的一种，对软件设计中的单元模块进行测试。

集成测试：在单元测试的基础上，对单元模块之间的连接和组装进行测试。

系统测试：在所有都考虑的情况下，对系统进行测试。

验收测试：第三方进行的确认软件满足需求的测试。

## 14. 黑盒测试和白盒测试常用的测试方法有哪些，举个例子？

黑盒有等价类划分法，边界分析法，因果图法和错误猜测法。

白盒有逻辑覆盖法，循环测试路径选择，基本路径测试。

例子：在一次输入多个条件的完整性查询中。利用等价类划分法则和边界分析法则，首 先利用等价类划分法，可以一个或多个结果是 OK 的测试用例，然后确认多个 NG 的测试用例， 然后利用边界值分析法，可以对结果分别是 OK 和 NG 的测试用例进行扩展和补充。

## 15. 简述黑盒测试和白盒测试的优缺点？

※ 黑盒测试的优点有：

1. 比较简单，不需要了解程序内部的代码及实现；
2. 与软件的内部实现无关；
3. 从用户角度出发，能很容易的知道用户会用到哪些功能，会遇到哪些问题；
4. 基于软件开发文档，所以也能知道软件实现了文档中的哪些功能；
5. 在做软件自动化测试时较为方便。

※ 黑盒测试的缺点有：

1. 不可能覆盖所有的代码，覆盖率较低，大概只能达到总代码量的 30%；
2. 自动化测试的复用性较低。

※ 白盒测试的优点有：

1. 帮助软件测试人员增大代码的覆盖率，提高代码的质量，发现代码中隐藏的问题。

※ 白盒测试的缺点有：

1. 程序运行会有很多不同的路径，不可能测试所有的运行路径；测试基于代码，只能测试开发人员做的对不对，而不能知道设计的正确与否，可能会漏掉一些功能需求；系统庞大时，测试开销会非常大。

## 16. 在没有产品说明书和需求文档的情况下能够进行黑盒测试的设计吗？

能，可以通过其他工作内容去替代产品说明书和需求文档

根据客户的功能点整理测试需求追溯表

根据开发人员的 Software Specification List 整理功能测试点

开展项目跨部门讨论会，主要整理对功能点的理解和认识

测试人员整理用例需求疑问提交项目组或者产品

项目内部的用例品胜

邮件客户代表确认部分争议问题

项目的 Demo 和部分已经开发的系统

参考同行业和竞争对手的类似产品

交叉模块之间的测试

咨询客户或相关者

## 17. 单元测试的策略有哪些，主要内容有哪些？

逻辑覆盖，循环覆盖，同行评审，桌前检查，代码走查，代码评审，静态数据流分析

### 18. 白盒测试逻辑覆盖有哪几种覆盖标准，覆盖率最高的是什么？

语句覆盖,分支覆盖,条件覆盖,路径覆盖,分支条件覆盖,覆盖率最高的是路径覆盖

### 19. Beta 测试和 Alpha 测试有什么区别？

大型通用软件，在正式发布前，通常需要执行 Alpha 和 Beta 测试，目的是从实际终端用户的使用角度，对软件的功能和性能进行测试，以发现可能只有最终用户才能发现的错误。

Alpha 测试是由一个用户在开发环境下进行的测试，也可以是公司内部的用户在模拟实际操作环境下进行的受控测试，Alpha 测试不能由程序员或测试员完成。Alpha 测试发现的错误，可以在测试现场立刻反馈给开发人员，由开发人员及时分析和处理。目的是评价软件产品的功能、可使用性、可靠性、性能和支持。尤其注重产品的界面和特色。Alpha 测试可以从软件产品编码结束之后开始，或在模块（子系统）测试完成后开始，也可以在确认测试过程中产品达到一定的稳定和可靠程度之后再开始。有关的手册（草稿）等应该在 Alpha 测试前准备好。

Beta 测试是软件的多个用户在一个或多个用户的实际使用环境下进行的测试。开发者通常不在测试现场，Beta 测试不能由程序员或测试员完成。因而，Beta 测试是在开发者无法控制的环境下进行的软件现场应用。在 Beta 测试中，由用户记下遇到的所有问题，包括真实的以及主管认定的，定期向开发者报告，开发者在综合用户的报告后，做出修改，最后将软件产品交付给全体用户使用。Beta 测试着重于产品的支持性，包括文档、客户培训和支持产品的生产能力。只有当 Alpha 测试达到一定的可靠程度后，才能开始 Beta 测试。由于 Beta 测试的主要目标是测试可支持性，所以 Beta 测试应该尽可能由主持产品发行的人员来管理。

## 五、 测试流程

### 20. 软件测试的基本流程有哪些？

需求分析、编写测试用例、评审测试用例、搭建环境、等待程序开发包、部署程序开发包、冒烟测试、执行具体的测试用例细节、Bug 跟踪处理回归测试、N 轮之后满足需求，测试结束

### 21. 测试结束的标准是什么？

第一类标准：测试超过了预定时间，则停止测试。

第二类标准：执行了所有的测试用例，但并没有发现故障，则停止测试。

第三类标准：使用特定的测试用例设计方案作为判断测试停止的基础

第四类标准：正面指出停止测试的具体要求，即停止测试的标准可定义为查出某一预订数目的故障。

第五类标准：根据单位时间内查出故障的数量决定是否停止测试。

### 22.软件测试的原则是什么？

1) 应当把“尽早地和不断地进行软件测试”作为软件开发者的座右铭。
2) 测试用例应由测试输入数据和对应的预期输出结果这两部分组成。
3) 程序员应避免检查自己的程序。
4) 在设计测试用例时，应包括合理的输入条件和不合理的输入条件。
5) 软件测试的原则
6) 充分注意测试中的群集现象。 经验表明，测试后程序中残存的错误数目与该程序中已发现的错误数目成正比。
7) 严格执行测试计划，排除测试的随意性。
8) 应当对每一个测试结果做全面检查。
9) 妥善保存测试计划，测试用例，出错统计和最终分析报告，为维护提供方便。

## 六、 用例设计

### 23.什么是测试用例，测试用例的基本要素？

测试用例是为某个特殊目标而编制的一组测试输入、执行条件以及预期结果，以便测试某个程序路径或核实是否满足某个特定需求。

测试用例的基本元素： 测试索引，测试环境，测试输入，测试操作，预期结果，评价标准。

### 24.描述测试用例设计的完整过程？

首先根据需求文档、概要设计、测试计划、测试方案细分出各功能模块的测试项

再根据各测试项，按照概要设计、详细设计以及测试方案中测试的覆盖率细分出测试子项

最后按照测试子项、根据测试用例的设计方法（因果图、边界值、等价类等的设计方法）书写测试用例。

注意

- 选用适合的用例管理工具（如 word，excel）
- 用例一定要及时更新（补充新的想法，删除过时的需求）
- 做好用例分级
- 做好用例评审，写用例之前可以征询相关人员的意见，如果评审通过可以参考其执行测试，如果未通过，需要继续修改直到通过为止。
- 可以考虑结对编写，这个是不错的主意
- 要全面，包括功能、性能、兼容性、安全性、易用性、容错性等等

- 注意把握适当的颗粒度

## 25.好的测试用例有哪些特点？

质量属性：

正确性：确保测试标题描述部分的内容正确性。

经济性：只为确定需要的目的设计相应的测试步骤。

可重复性：自我一致性，即不管谁执行此用例，结果一样。

适应性：既能适应短期需要，又能考虑长远需要。

可追踪性：用例能追踪到一个具体的需求。

自我清理性：单个用例不会影响整个测试环境，即用例执行完了可以恢复原有的测试环境。

结构化和可测试性

含有规范的测试标题和编号。

含有一个确定的测试某一个特定需求的目的。

含有关于测试方法的描述。

指定条件信息-环境、数据、预置的条件测试、安全入口等。

含有操作步骤和预期结果。

陈述任何辅助证据，例如截图报告并确保这些东西妥善保存。

确保测试环境的干净（即用例不会影响整个环境）。

描述时使用主动语气结构。

操作步骤不要超过 15 步。

确保单个用例测试执行时用时不超过 20 分钟。

自动化脚本用例添加必要的注释，比如目的、输入和期望结果。

如果可能，建议提供可选择性的预置条件测试。

用例之间的先后顺序是否跟业务流程一致，即用例在业务流程中的彼此顺序关系是否合理。

配置管理：

采用命名和编号规范归档。

保存为特定的格式，文件类型。

用例版本是否与当前被测试软件版本一致（对应）。

包含用例需要的相应测试对象，如特定数据库。

存档阅读。

存档时按角色控制访问方式

当网络备份时存档。

离线归档。

## 26.写测试用例时要注意什么问题

1、复用率：如果随着产品不停得升级，需要设计的详细些，追求一劳永逸；仅使用一两次，则没有必要设计的过于详细；

2、项目进展：项目时间如果允许可以设计的详细些，反之则能执行即可；

3、使用对象：测试用例如果供多人使用，尤其让后参加测试的工程师来执行，则需要设计的详细些。

4、用例的冗余

5、操作步骤要细分简明，可执行

## 27.如何在有限的情况下提高测试效率，保证产品的上线质量？

1、一个详细合理的详细的测试计划

2、测试尽早的介入项目，连接项目的业务需求，做好测试的前期准备

3、对测试项目前景充满信心，调整最佳心态，保持愉悦的工作心情

4、提高测试接受的标准，减少测试版本的送测次数

## 28.如何降低漏测率

1、需求评审

2、梳理需求，尽早与开发人员、需求人员进行需求确认，统一不同角色对需求的认识

3、用例设计及评审

4、测试执行

5、bug 回归

6、发布前的功能回归

## 29.测试用例的基本设计方法

1、等价类划分法

2、边界值分析法

3、错误推断法

4、因果图判定表法

5、正交实验法

6、流程法

7、场景法

### 30.测试为什么要写测试用例

1、深入了解需求的过程

2、测试执行的指导

3、规划测试数据的准备

4、反应测试进度

5、举一反三发现隐藏缺陷

6、分析缺陷标准

## 七、 缺陷 bug

### 31.什么是缺陷报告，缺陷报告的作用，缺陷报告的要点

(1)缺陷报告是描述软件缺陷现象和重现步骤的集合。软件缺陷报告 Software Bug Report(SBR)或软件问题报告 software Problem Report(SPR)。

(2)缺陷报告是软件测试人员的工作成果之一，体现软件测试的价值缺陷报告可以把软件存在的缺陷准确的描述出来，便于开发人员修正缺陷报告可以反映项目/产品当前的质量状态，便于项目整体进度和质量控制软件测试缺陷报告是软件测试的输出成果之一，可以衡量测试人员的工作能力。

(3)标题(Title)简洁、准确、完整、反映缺陷本质、方便查询前缀+标题正文，标题正文采用结果和动作，或者现象和位置的方式表达；步骤(Steps)可复现、完整、简洁、准确按数字编号；实际结果(Actual results)准确、详细描述软件的现象和特征；期望结果(Expected results)准确、丰富、有理有据；平台(Platforms)准确；截图(Sereenshots)准确反映缺陷特征；注释(Notes)关于缺陷的辅助说明

### 32.软件测试缺陷报告的 5C 原则

Correct（准确）：每个组成部分的描述准确，不会引起误解；

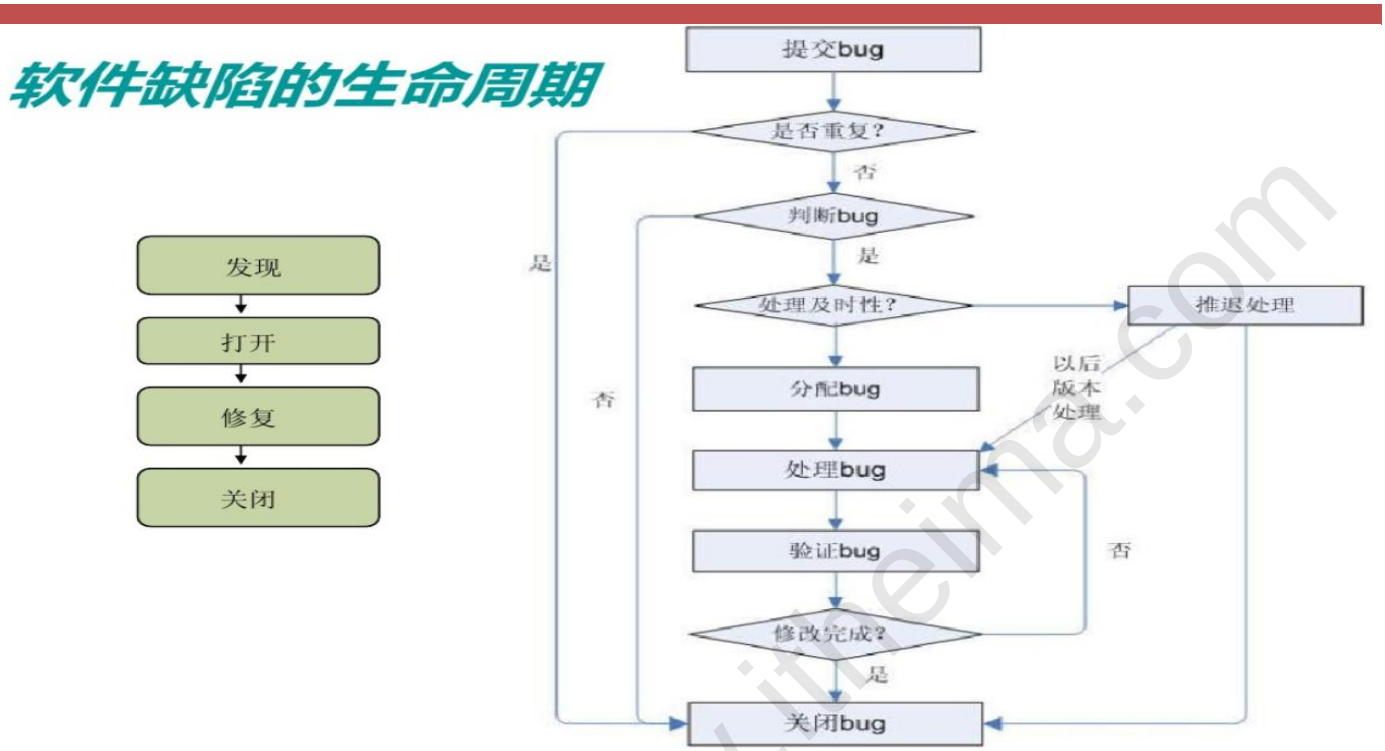
Clear（清晰）：每个组成部分的描述清晰，易于理解；

Concise（简洁）：只包含必不可少的信息，不包括任何多余的内容；

Complete（完整）：包含复现该缺陷的完整步骤和其他本质信息；

Consistent（一致）：按照一致的格式书写全部缺陷报告。

### 33.软件缺陷的生命周期？




测试人员提交新的 Bug 入库，错误状态为 New。 高级测试人员验证错误，如果确认是错误，分配给相应的开发人员，设置状态为 Open。如果不是错误，则拒绝，设置为 Declined(拒绝)状态。开发人员查询状态为 Open 的 Bug，如果不是错误，则置状态为 Declined；如果是 Bug 则修复并置状态为 Fixed。不能解决的 Bug，要留下文字说明及保持 Bug 为 Open 状态。对于不能解决和延期解决的 Bug，不能由开发人员自己决定，一般要通过某种会议（评审会）通过才能认可。 测试人员查询状态为 Fixed 的 Bug，然后验证 Bug 是否已解决，如解决置 Bug 的状态为 Closed，如没有解决置状态为 Reopen。

## 34. 缺陷描述（报告单）中应该包括哪些内容？

缺陷的标题，简要描述。缺陷的类型。缺陷的详细步骤描述。缺陷的实际结果。期望结果。有的缺陷需要上传截图，日志信息。缺陷的等级。缺陷指派给开发同事。（开发主管）

## 35. 如何提高缺陷的记录质量？

通用 UI 要统一、准确；尽量使用业界惯用的表达术语和表达方法；使用业界惯用的表达术语和表达方法，保证表达准确，体现专业化；每条缺陷报告只包括一个缺陷；不可重现的缺陷也要报告；明确指明缺陷类型；明确指明缺陷严重等级和优先等级；描述 (Description) ，简洁、准确，完整，揭示缺陷实质，记录缺陷或缺陷出现的位置；短行之间使用自动数字序号，使用相同的字体、字号、行间距； 每一个步骤尽量只记录一个操作；确认步骤完整，准确，简短；根据缺陷，可选择是否进行图象捕捉； 检查拼写和语法缺陷；尽量使用短语和短句，避免复杂句型句式；缺陷描述内容。

## 36. 如果一个缺陷被提交后，开发人员认为不是问题，怎么处理？

a)首先，将问题提交到缺陷管理库里面进行备案。

b)然后，要获取判断的依据和标准：

- v.根据需求说明书、产品说明、设计文档等，确认实际结果是否与计划有不一致的地方，提供缺陷是否确认的直接依据；
- vi.如果没有文档依据，可以根据类似软件的一般特性来说明是否存在不一致的地方，来确认是否是缺陷；
- vii.根据用户的一般使用习惯，来确认是否是缺陷；
- viii.与设计人员、开发人员和客户代表等相关人员探讨，确认是否是缺陷；

c)合理的论述，向测试经理说明自己的判断的理由，注意客观、严谨，不掺杂个人情绪。

d)等待测试经理做出最终决定，如果仍然存在争议，可以通过公司政策所提供的渠道，向上级反映，并有上级做出决定。

### 37.缺陷的优先级划分和描述

一般来说按照下面的来分，具体的是由每个公司而定。

软件缺陷有四种级别，分别为：致命的(Fatal)，严重的(Critical)，一般的(Major)，微小的(Minor)。

A 类—致命的软件缺陷(Fatal):造成系统或应用程序崩溃、死机、系统挂起，或造成数据丢失，主要功能完全丧失，导致本模块以及相关模块异常等问题。如代码错误，死循环，数据库发生死锁、与数据库连接错误或数据通讯错误，未考虑异常操作，功能错误等

B 类—严重错误的软件缺陷（critical）：系统的主要功能部分丧失、数据不能保存，系统的次要功能完全丧失。问题局限在本模块，导致模块功能失效或异常退出。如致命的错误声明，程序接口错误，数据库的表、业务规则、缺省值未加完整性等约束条件

C 类—一般错误的软件缺陷（major）：次要功能没有完全实现但不影响使用。如提示信息不太准确，或用户界面差，操作时间长，模块功能部分失效等，打印内容、格式错误，删除操作未给出提示，数据库表中有过多的空字段等

D 类—较小错误的软件缺陷（Minor）:使操作者不方便或遇到麻烦，但它不影响功能过的操作和执行，如错别字、界面不规范（字体大小不统一，文字排列不整齐，可输入区域和只读区域没有明显的区分标志），辅助说明描述不清楚

E 类- 建议问题的软件缺陷（Enhancemental）：由问题提出人对测试对象的改进意见或测试人员提出的建议、质疑。

## 八、 测试实例

### 38.一个有广告的纸杯子，请设计测试用例？

测试项目：杯子

需求测试：查看杯子使用说明书

界面测试：查看杯子外观

功能度：用水杯装水看漏不漏；水能不能被喝到

安全性：杯子有没有毒或细菌

可靠性：杯子从不同高度落下的损坏程度

可移植性：杯子在不同的地方、温度等环境下是否都可以正常使用

兼容性：杯子是否能够容纳果汁、白水、酒精、汽油等

易用性：杯子是否烫手、是否有防滑措施、是否方便饮用

用户文档：使用手册是否对杯子的用法、限制、使用条件等有详细描述

疲劳测试：将杯子盛上水（案例一）放 24 小时检查泄漏时间和情况；盛上汽油（案例二）放 24 小时检查泄漏时间和情况等

压力测试：用根针并在针上面不断加重量，看压强多大时会穿透

跌落测试: 杯子加包装(有填充物),在多高的情况摔下不破损

震动测试: 杯子加包装(有填充物),六面震动,检查产品是否能应对恶劣的铁路\公路\航空运输

基本功能测试（逻辑功能测试）。

（1）硬度：是否达到设计标准。

装载能力：在杯子内分别装入少量的、半杯的、满杯的，看其装载量是否达到设计标准。

装载种类：开水（是否产生异味）、温水、冷水、冰水、咖啡…

（2）界面测试（UI 测试）。

看其形状、大小设计是否适合人方便拿起。

外观是否吸引人（广告嘛），赏心悦目。

带广告的图案沾水受是否掉色、模糊。

（3）易用性测试。

看其形状、大小设计是否适合人方便拿起。

残疾人士用此杯去喝水的容程度。

杯子设计是否上大下小，在运输过程中可以套在一起有效利用空间，在使用时也容易拿开。

（4）稳定性测试（24 X 7 测试）。装入液体后记录其多少以后漏水。

（5）安全性测试。杯子所用的材料（包括纸基、涂层和广告颜料）是否符合食品卫生标准，在内外温度等环境因素下是否会与所盛各种饮料相反应，而产生对人体有害的物质。

（6）本地化测试。为国际化和本地化的需要，广告图案和文字是否在政治、宗教和文化方面具有广泛的适用性。

（7）对设计的改进建议。“如果是一次性杯子，能否标示已使用（比如变色）”和“杯子是否有使用者标贴（多人使用时防止混淆）”。

## 39.一个身份证号码输入框，怎么设计用例？

校验身份证号规则的有效性（包括地址码、生日期码、顺序码和校验码

校验 15 位身份证号和 18 位身份正好都是可用的

校验末位是 X 的情况

校验不足 15 位、16-17 位和大于 18 位的情况

如果是必输项，校验不输入的时候会不会有正确的提示

如果不是必输项，则要校验不输入的时候流程能否正常进行

校验输入非数字的情况，是否会有正确提示信息（包括大小写字母、汉字、特殊字符和标点符号）

校验输入全角的数字的时候，系统是否会识别（这个得根据需求确定是否可以使用全角的数字）

## 40.登录功能怎么设计测试用例？

具体需求：

有一个登录页面，有一个账号和一个密码输入框，一个提交按钮。

此题的考察目的：

1、了解需求（测什么都是从了解需求开始）；

2、是否有设计 Test Case 的能力

3、是否熟悉各种测试方法；

4、是否有丰富的 Web 测试经验；

5、是否了解 Web 开发；

了解需求：

1、登录界面应该是弹出窗口式的，还是直接在网页里面；

2、账号长度和密码的强度（比如需要多少位、大小写敏感、特殊字符混搭等）；

3、界面美观是否有特殊要求？（即是否要进行 UI 测试）；

4、….

用例设计：

测试需求分析完成后，开始用例设计，主要可以从以下几个方面考虑：

功能测试(Function Test)

1、输入正确的账号和密码，点击提交按钮，验证是否能正确登录。（正常输入）

2、输入错误的账号或者密码，验证登录会失败，并且提示相应的错误信息。（错误校验）

3、登录成功后能否跳转到正确的页面（低）

4、账号和密码，如果太短或者太长，应该怎么处理（安全性，密码太短时是否有提示）

5、账号和密码，中有特殊字符（比如空格），和其他非英文的情况（是否做了过滤）

6、记住账号的功能

7、登录失败后，不能记录密码的功能

8、账号和密码前后有空格的处理

9、密码是否加密显示（星号圆点等）

10、牵扯到验证码的，还要考虑文字是否扭曲过度导致辨认难度大，考虑颜色（色盲使用者），刷新或换一个按钮是否好用

11、登录页面中的注册、忘记密码，登出用另一帐号登录等链接是否正确

12、输入密码的时候，大写键盘开启的时候要有提示信息。

13、什么都不输入，点击提交按钮，看提示信息。（非空检查）

界面测试(UI Test)

1、布局是否合理，2 个 Testbox 和一个按钮是否对齐

2、Testbox 和按钮的长度，高度是否符合要求

3、界面的设计风格是否与 UI 的设计风格统一

4、界面中的文字简洁易懂，没有错别字。

性能测试(Performance Test)

1、打开登录页面，需要几秒

2 、输入正确的账号和密码后，登录成功跳转到新页面，不超过 5 秒

安全性测试(Security Test)

1、登录成功后生成的 Cookie 是否有 HttpOnly(降低脚本盗取风险)

2、账号和密码是否通过加密的方式，发送给 Web 服务器

3、账号和密码的验证，应该是用服务器端验证，而不能单单是在客户端用 javaScript 验证

4、账号和密码的输入框，应该屏蔽 SQL 注入攻击

5、账号和密码的输入框，应该禁止输入脚本（防止 XSS 攻击）

6、错误登录的次数限制（防止暴力破解）

7、考虑是否支持多用户在同一机器上登录；

8、考虑一用户在多台机器上登录

可用性测试(Usability Test)

1、是否可以全用键盘操作，是否有快捷键

2、输入账号，密码后按回车，是否可以登录

3、输入框是否可以以 Tab 键切换

兼容性测试（Compatibility Test）

1、主流的浏览器下能否显示正常已经功能正常（IE6~11, FireFox, Chrome, Safari 等 ）

2、不同的平台是否能正常工作，比如 Windows, Mac

3、移动设备上是否正常工作，比如 iPhone, Android

4、不同的分辨率

本地化测试 （Localization Test）

1、不同语言环境下，页面的显示是否正确。

软件辅助性测试 （Accessibility Test）

软件辅助功能测试是指测试软件是否向残疾用户提供足够的辅助功能

1、高对比度下能否显示正常（视力不好的人使用）

## 41.移动端和 web 端测试有什么区别

单纯从功能测试的层面上来讲的话，APP 测试、web 测试 在流程和功能测试上是没有区别的。根据两者载体不一样，则区别如下：

系统结构方面

web 项目，b/s 架构，基于浏览器的；web 测试只要更新了服务器端，客户端就会同步会更新。

app 项目，c/s 结构的，必须要有客户端；app 修改了服务端，则客户端用户所有核心版本都需要进行回归测试一遍。

性能方面

web 项目 需监测 响应时间、CPU、Memory

app 项目 除了监测 响应时间、CPU、Memory 外，还需监测 流量、电量等

兼容方面

（1）web 项目：

1. 浏览器（火狐、谷歌、IE 等）
2. 操作系统（Windows7、Windows10、Linux 等）

（2）app 项目：

1. 设备系统:iOS（ipad、iphone）、Android（三星、华为、联想等） 、Windows（Win7、Win8）、 OSX（Mac）
2. 手机设备可根据 手机型号、分辨率不同

相对于 Wed 项目，APP 有专项测试

1. 干扰测试：中断，来电，短信，关机，重启等
2. 弱网络测试（模拟 2g、3g、4g，wifi 网络状态以及丢包情况）；网络切换测试（网络断开后重连、3g 切换到 4g/wifi 等）
3. 安装、更新、卸载

安装：需考虑安装时的中断、弱网、安装后删除安装文件等情况

卸载：需考虑 卸载后是否删除 app 相关的文件

更新：分强制更新、非强制更新、增量包更新、断点续传、弱网状态下更新

4. 界面操作：关于手机端测试，需注意手势，横竖屏切换，多点触控，前后台切换

5. 安全测试：安装包是否可反编译代码、安装包是否签名、权限设置，例如访问通讯录等

6. 边界测试：可用存储空间少、没有 SD 卡/双 SD 卡、飞行模式、系统时间有误、第三方依赖（QQ、微信登录）等

7. 权限测试：设置某个 App 是否可以获取该权限，例如是否可访问通讯录、相册、照相机等

测试工具方面

自动化工具：APP 一般使用 Appium; Web 一般使用 Selenium

性能测试工具：APP 一般使用 JMeter; Web 一般使用 LR、JMeter

## 42.测试一个 C/S 客户端时，需要考虑的因素

客户端安装测试

客户端升级测试

客户端可维护性测试

（1）个体的客户端应用以“分离的”模式被测试——不考虑服务器和底层网络的运行；

（2）客户端软件和关联的服务器端应用被一起测试，但网络运行不被明显的考虑；

（3）完整的 C/S 体系结构，包括网络运行和性能被测试。

应用功能测试——客户端应用被独立地执行，以揭示在其运行中的错误。

服务器测试——测试服务器的协调和数据管理功能，也考虑服务器性能（整体反映时间和数据吞吐量）。

数据库测试——测试服务器存储的数据的精确性和完整性，检查客户端应用提交的事务，以保证数据被正确地存储、更新和检索。

事务测试——创建一系列的测试以保证每类事务被按照需求处理。测试着重于处理的正确性，也关注性能问题。

网络通信测试——这些测试验证网络节点间的通信正常地发生，并且消息传递、事务和相关的网络交通无错的发生

## 43.测试电梯，请详细描述

**如果给你一台电梯，请问你如何测试它，分析如下：**

1.功能：上升、下降、停止、开门、关门、梯内电话、灯光、指示灯等；

2.性能：速度、反应时间、关门时间等；

3.压力：超载、尖锐物碰撞电梯壁等；

4.安全：停电、报警装置、轿箱停靠位置、有人扒门时的情况等；

5.可用性：按键高度、操作是否方便、舒适程度等；

6.UI：美观程度、光滑程度、形状、质感等；

7.稳定性：长时间运行情况等；

8.兼容性：不同电压是否可工作、不同类型电话是否可安装等。其实在简单分析的过程中，发现许多东西根本测试不全，比如电话、灯光、材质、调度程序、可维修性等，当发现在一个用例中无法说清楚时，这些应该拆分开来分别测试。可以告诉主考官，你需要模块化地测试电话、灯光等。再有在一起的组装测试。

**下面是详细的测试点：**

需求测试：查看电梯使用说明书、安全说明书等

界面测试：查看电梯外观

功能测试：1.测试电梯能否实现正常的上升和下降功能。

2.电梯的按钮是否都可以使用。

3.电梯门的打开，关闭是否正常。

4.报警装置是否可用。

5.与其他电梯之间是否协作良好。

6.通风状况如何。

7.突然停电时的情况。

8.上升途中的响应。1）电梯本来在 1 楼，如果有人按 18 楼，那么电梯在上升到 5 楼的时候，有人按了 10 楼，这时候是否会在 10 楼先停下来

2）电梯下降到 10 层时显示满员，此时若 8 层有人等待电梯，是否在 8 层停。

9.是否有手机信号

可靠性测试：

1.门关上的一刹那出现障碍物。

2.同时按关门和开门按钮。

3.点击当前楼层号码

4.多次点击同一楼层号码

5.同时按上键和下键

易用性：电梯的按钮的设计符合一般人的习惯吗

用户文档：使用手册是否对电梯的用法、限制、使用条件等有详细的描述

压力测试：1.看电梯的最大承重量，在负载过重时报警装置是否有提醒

稳定性测试：看垫底在最大负载下平行运行的最长时间

## 44.对一只圆珠笔进行测试

1. 界面测试，无论我们做那类软件（嵌入式别提），只要给用户有看到的东西，从测试的角度，就要考虑界面测试，这个呢，现在针对微软的产品，某公司开发了一套界面检查表，我这里有一份，想要可以找我

界面测试测什么，怎么测呢？针对这个问题我是这样回答的，印刷在产品上的图片，文字，这可能涉及不同的东西，有圆珠笔厂家的信息，也有针对不同用户的信息（譬如小孩子喜欢颜色搭配多一点的，而成人用稳重的产品

等），可能涉及的还有人的审美观，你圆珠笔色彩搭配之类的

2. 功能测试，这是我们测试的重点，也是客户针对某家公司产品给出满意度的参考点，圆珠笔功能主要是书写，这里面涉及一个功用方面的焦点——书写的快慢程度，也就是流利不流利的问题（这涉及笔芯的材质问题）

针对这方面的测试，个人认为应从以下几点

a. 材质问题，这涉及程序员和用户之间的关系，两者利益均有，程序员考虑成本问题，用户考虑污染问题，也就是说制作圆珠笔的材料与环境的问题，厂商考虑价格因素，用户考虑环境因素以及安全性因素

这就把安全性测试给说出来了，大的方面因为笔油材质的问题，和使用者之间的健康问题有联系，

要测小的方面，笔油的速率，以及书写后是否马上可以涂抹，可否修改，这都涉及安全性的问题

b. 性能问题，温度，湿度，气压对笔芯产生不同的影响

3. 安全性问题

测试不同的高度，笔身做自由落体损坏程度

4. 兼容性问题

不同的笔筒和笔芯之间的互相兼容

5. 强度测试

弹簧在不同的压力之下，承受变形的程度

6. 在金山面试时候，考官特意问我针对笔芯那个米珠如何测试

或者

1、界面测试

界面测试也就是对其外表先进行判断。

尺寸是否适合用户使用？用户需要的是什么样的尺寸，小孩和成年人使用的尺寸是有区别的；

色彩搭配是否合理？形状是否美观？

是否方便携带和存放？

笔芯颜色是否与客户要求一致？

笔身印的 logo 或者文字是否这么正确

2、功能测试

笔筒开合；

笔芯替换；

出墨快慢；

笔头出墨粗细；

是不是可操作性签字笔；

3、性能测试

笔芯的寿命；

笔墨的气味；

写过的字用纸水浸透后，笔墨是否会晕开

压力测试：笔尖在多大压力范围内可以正常写字，不能正常出墨，太重损坏笔尖或纸张；

笔壳能在多大压力范围内正常使用？成人用力太重掰断笔壳，掉到地上易摔，能在纸上写出清晰的字

4、性能测试

握笔的地方纹路是否会硌手或太滑；

书写的流畅度；

写出的墨水多久能干；

高温和低温环境对笔芯出墨和笔壳的影响；

长时间不盖笔套，或笔盖盖多长时间不用，会不会对笔下次写字有影响

5、安全测试

笔墨是否有易燃性；

笔墨是否对皮肤有害；

笔杆折断，材质是否容易刮伤手；

误食笔芯是否会引起中毒（有小孩或者有人喜欢咬笔头）

6、兼容性测试

笔壳和笔芯是否能够很好的适应主流签字笔尺寸；

这个笔芯的笔尖如果损坏，换上其他的笔芯的笔尖是否能用；

这个笔芯的笔墨如果用完，换上其他笔芯的笔墨是否可以使用；

笔的笔墨如果在其他笔的笔墨上写字是否可以成功覆盖

7、其他测试

（1）比较测试

与其他品牌签字笔比较，优劣在哪些地方？

（2）场景测试

笔从高处摔到地上，笔尖是否会摔坏；

倒着写，是否可以写出很多字来；

扔到水里，笔墨会不会一直晕开；

笔在粗糙的纸上是否能写出字…

## 45.测试一个网上购物的购物车

界面测试：

·打开页面后，页面的布局是否合理，显示是否完整；

·鼠标浮动在购物车按钮，迷你购物车界面显示是否正常；

·不同卖家的商品在不同的 table 区域显示，区分明显；

·页面的 tooltips 能正常显示；

功能测试：

·所有页面链接功能正常，可以点击到正确页面；

·页面关联本地软件阿里旺旺的 icon 点击后，能打开软件；

·从商品信息页面添加的商品能显示在购物车中；

·购物车页面打开的同时，在其他页面添加了商品，购物车页面刷新后，新的商品能显示；

·若未登录，点击购物车，则提示用户输入用户名和密码，或者提示其他的非注册用户购物方式；

·商品未勾选的状态下，结算按钮是灰色无法点击的；

·勾选商品后，已选商品的总价会显示，结算按钮变高亮可点击工作；

·勾选商品，点击结算按钮后，进入确认订单信息页面；

·购物车页面中，可以对添加的商品信息做信息的修改，并自动保存成功；

·卖家在线的时候，旺旺 icon 高亮，反之，灰色；

·购物车有商品降价或者库存告急的，那么点击对应的 tab，降价或者告急商品会归类后显示；

·购物车能添加的商品种类是有数量上限的；

·不要的商品，可以删除；

（其他特有的功能不做赘述，只讨论常见通用功能）

若商品已经失效，购物车的商品是否可以继续结算

已进入支付界面但支付未成功，重新进入购物车，又重新添加了一些物品，则原有的物品是否能正确保留；（感觉这个还挺关键，经常是没完成支付，又添加了一些物品，最后再一起支付）

性能测试：

·打开购物车页面要多久；

可用性测试：

·快捷键功能知否支持

兼容测试：

·不同浏览器上的测试功能是否正常；

·app 上测试

## 46.请以微信点赞，功能点进行测试

1. 功能测试

考虑功能是否符合预期

2. 接口

考虑各内部和外部的接口，比如朋友圈客户端和服务端的交互接口的功能。朋友圈点赞功能和消息提示功能的接口（点了赞之后对应的朋友收到提示信息）

3. 平台

手机版 pad 版 web 版

4. 用户操作场景

测试用户常用场景，比如：用户打开微信看到十条消息提示，点击后进入朋友圈界面显示了“谁谁谁点了赞”

5. 速度、延迟

6. 性能测试

模拟一些多用户并发操作的场景

7. 安全

## 47.搜索框怎么测

增：

搜索功能分为简单搜索和高级搜索

简单搜索：

先展示百度的简单搜索界面，仅供参考：


界面测试

- 搜索框 UI 显示正常，布局合理；
- 搜索页面布局合理，无错别字；
- 搜索出的结果展示，布局合理；
- 已查看过的结果链接，链接的眼神要灰化处理，和没有点击过的结果链接区分；
- 结果数量庞大时，页面的分页布局合理；

功能测试

- 链接测试：页面上的链接都可连接至正确的页面
- 搜索历史内容记录，便于查找检索过的内容
- 搜索内容联想输入，便于用户搜索的便捷与准确性
- 搜索功能测试（重点）
    - 搜索内容为空，验证系统如何处理
    - 搜索内容为空格，查看系统如何处理
    - 边界值验证，在允许的字符串范围内外，验证系统的处理
    - 超长字符串的输入，系统是否会截取允许的长度来检索结果

- 合法的字符串长度后，加空格，验证检索结果
- 多个关键词中间加入空格，tab，逗号后，验证系统的结果是否正确
- 验证每种合法的输入，结果是否正确
- 是否支持检索内容的复制、粘贴、编辑等操作
- 是否支持回车键搜索
- 多次输入相同的内容，查看系统每次检索的结果是否正确，相同
- 特殊字符，转义符，html 脚本等需作处理
- 敏感词汇，提示用户无权限等信息
- 输入的内容，是否支持快捷键操作等
- 只能输入允许的字符串长度

安全性测试（没有做过，只能列出一些简单的注意点）

- 脚本的禁用
- SQl 注入，检索 sql select 语句等
- 敏感内容的检索是禁止的
- 特殊字符的检索
- 被删除、加密、授权的数据，不允许被查出来的，是否有安全控制设计

兼容性测试

- 多平台 windows，mac
- 移动平台 ios，android
- 多浏览器 FF，Chrome，IE，国内主流浏览器等

性能测试

- 搜索页面打开的速度是否满足设计要求
- 搜索出结果消耗的时间，是否满足设计要求

本地化测试

- 登录时，自动切换至相应语言国家的搜索主页

高级搜索：

先展示百度的高级搜索页面，仅供参考：


场景法测试，主要是为了验证搜索结果的正确性：

| 场景编号 | 场景描述 | 预期结果 |
| --- | --- | --- |
| 场景一 | 页面检查 | 正确 |
| 场景二 | 默认条件搜索 | 查询结果正确 |
| 场景三 | 修改可选条件搜索 | 查询结果正确 |
| 场景四 | 修改输入条件搜索 | 查询结果正确 |
| 场景五 | 修改区间条件搜索 | 查询结果正确 |
| 场景六 | 组合可选、输入条件搜索 | 查询结果正确 |
| 场景七 | 操作后检查搜索条件及查询结果 | 查询结果正确 |
| 场景八 | 错误、空记录搜索 | 查询结果为空 |

每个场景，对应了一种高级搜索活动从开始到退出搜索的完整过程

# 第三章　Linux 基础

## 48. 查看占用 CPU 使用率最高的进程？

ps -aux | sort -k3nr | head -K

## 49. 如何查看一个文件的末尾 50 行？

查看/etc/profile 的前 10 行内容，应该是：

# head -n 10 /etc/profile

查看/etc/profile 的最后 50 行内容，应该是：

# tail -n 50 /etc/profile

## 50. 如何过滤文件内容中包含”ERROR“的行？

grep "ERROR" file_name

cat file_name | grep "ERROR"

## 51. 查看某端口号？

netstat -anp | grep port_number

## 52. 查看某进程号？

ps -ef | grep ps_name

ps -ef | grep ps_number

## 53. rep 和 find 的区别？grep 都有哪些用法？

## 54. 查看 IP 地址？

ifconfig

## 55. 创建和删除一个多级目录？

mkdir -p ./a/b

rm -rf ./a

## 56.在当前用户家目录中查找 haha.txt 文件？

find ~/ -name haha.txt

## 57.如何查询出 tomcat 的进程并杀掉这个进程，写出 linux 命令？

ps -ef | grep tomcat

kill -9 tomcat_port

## 58.动态查看日志文件？

tail -f log_file

## 59.查看系统硬盘空间的命令？

df -aTh

## 60.查看当前机器 listen 的所有端口？

netstat -tlnp

## 61.把一个文件夹打包压缩成.tar.gz 的命令，以及解压拆包.tar.gz 的命令？

tar zcvf xxx.tar.gz file tar zxvf xxx.tar.gz

## 62.Xshell 工具如果想要实现从服务器上传或者下载文件的话,可以在服务器上安装什么包？

lrzsz

## 63.以/etc/passwd 的前五行内容为例，提取用户名？

cat /etc/passwd | head -n 5 | cut -d : -f 1

## 64.在 linux 中 find 和 grep 的区别？

Linux 系统中 grep 命令是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹配的行打印出来。grep 全称是 Global Regular Expression Print，表示全局正则表达式版本，它的使用权限是所有用户。

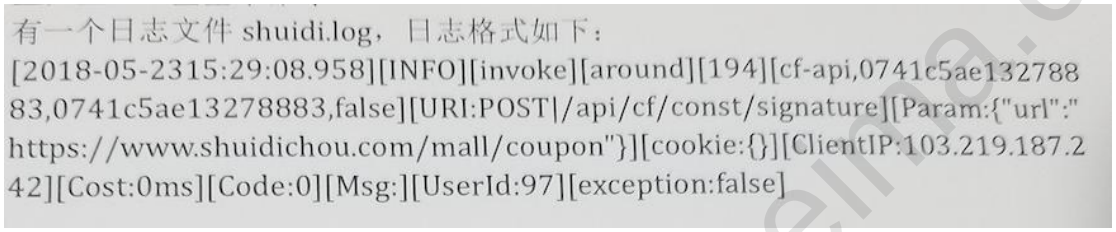
linux 下的 find：

功能：在目录结构中搜索文件，并执行指定的操作。此命令提供了相当多的查找条件，功能很强大。

语法：find 起始目录寻找条件操作说明：find 命令从指定的起始目录开始，递归地搜索其各个子目录，查找满足寻找条件的文件并对之采取相关的操作。

简单点说说，grep 是查找匹配条件的行，find 是搜索匹配条件的文件。

## 65.查看日志题


有一个日志文件 shuidi.log，日志格式如下：
[2018-05-2315:29:08.958][INFO][invoke][around][194][cf-api,0741c5ae132788
83,0741c5ae13278883,false][URI:POST|/api/cf/const/signature][Param:{"url":"
https://www.shuidichou.com/mall/coupon"}][cookie:{}][ClientIP:103.219.187.2
42][Cost:0ms][Code:0][Msg:][UserId:97][exception:false]

1. lang.nullpointerexception 这个错误常见原因有哪些？
2. LINUX 基本命令

1）怎么查出 shuidi.log 中最近一条 nullpointerexception，并统计 nullpointerexception 的日志行数；

2）查出 userid=97 这个用户的错误请求日志；

# 第四章　Mysql 基础

## 一、 基础知识

### 1. 什么是数据库？

数据库(Database)是按照数据结构来组织、存储和管理数据的仓库

### 2. 什么是关系型数据库，主键，外键，索引分别是什么？

关系型数据库是由多张能互相联接的二维行列表格组成的数据库

主关键字(primary key)是表中的一个或多个字段，它的值用于唯一地标识表中的某一条记录

外键表示了两个关系之间的相关联系。以另一个关系的外键作主关键字的表被称为主表，具有此外键的表被称为主表的从表。外键又称作外关键字

在关系数据库中，索引是一种单独的、物理的对数据库表中一列或多列的值进行排序的一种存储结构，它是某个表中一列或若干列值的集合和相应的指向表中物理标识这些值的数据页的逻辑指针清单

### 3. 表的连接查询方式有哪些，有什么区别？

交叉连接即笛卡儿乘积，是指两个关系中所有元组的任意组合

使用内连接时，如果两个表的相关字段满足连接条件，就从这两个表中提取数据并组合成新的记录

自连接是一种特殊的内连接，它是指相互连接的表在物理上为同一张表，但可以在逻辑上分为两张表

外连接是只限制一张表中的数据必须满足连接条件，而另一张表中的数据可以不满足连接条件的连接方式

### 4. SQL 的 select 语句完整的执行顺序？

1、from 子句组装来自不同数据源的数据；

2、where 子句基于指定的条件对记录行进行筛选；

3、group by 子句将数据划分为多个分组；

4、使用聚集函数进行计算；

5、使用 having 子句筛选分组；

6、计算所有的表达式；

7、select 的字段；

8、使用 order by 对结果集进行排序。

## 5. 说一下 Mysql 数据库存储的原理？

储存过程是一个可编程的函数，它在数据库中创建并保存。它可以有 SQL 语句和一些特殊的控制结构组成。当希望在不同的应用程序或平台上执行相同的函数，或者封装特定功能时，存储过程是非常有用的。数据库中的存储过程可以看做是对编程中面向对象方法的模拟。它允许控制数据的访问方式。存储过程通常有以下优点：

1、存储过程能实现较快的执行速度

2、存储过程允许标准组件是编程。

3、存储过程可以用流程控制语句编写，有很强的灵活性，可以完成复杂的判断和较复杂的运算。

4、存储过程可被作为一种安全机制来充分利用。

5、存储过程能够减少网络流量

## 6. 事务的特性？

1、原子性(Atomicity)：事务中的全部操作在数据库中是不可分割的，要么全部完成，要么均不执行。

2、一致性(Consistency)：几个并行执行的事务，其执行结果必须与按某一顺序串行执行的结果相一致。

3、隔离性(Isolation)：事务的执行不受其他事务的干扰，事务执行的中间结果对其他事务必须是透明的。

4、持久性(Durability)：对于任意已提交事务，系统必须保证该事务对数据库的改变不被丢失，即使数据库出现故障

## 7. 数据库索引？

数据库索引，是数据库管理系统中一个排序的数据结构，以协助快速查询、更新数据库表中数据。索引的实现通常使用 B_TREE。B_TREE 索引加速了数据访问，因为存储引擎不会再去扫描整张表得到需要的数据；相反，它从根节点开始，根节点保存了子节点的指针，存储引擎会根据指针快速寻找数据。

## 8. 数据库怎么优化查询效率？

1、储存引擎选择：如果数据表需要事务处理，应该考虑使用 InnoDB，因为它完全符合 ACID 特性。如果不需要事务处理，使用默认存储引擎 MyISAM 是比较明智的

2、分表分库，主从。

3、对查询进行优化，要尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索引

4、应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表扫描

5、应尽量避免在 where 子句中使用!= 或<>操作符，否则将引擎放弃使用索引而进行全表扫描

6、应尽量避免在 where 子句中使用 or 来连接条件，如果一个字段有索引，一个字段没有索引，将导致引擎放弃使用索引而进行全表扫描

7、Update 语句，如果只更改 1、2 个字段，不要 Update 全部字段，否则频繁调用会引起明显的性能消耗，同时带来大量日志

8、对于多张大数据量（这里几百条就算大了）的表 JOIN，要先分页再 JOIN，否则逻辑读会很高，性能很差。

## 9. 你用的 Mysql 是哪个引擎，各引擎之间有什么区别？

主要 MyISAM 与 InnoDB 两个引擎，其主要区别如下：InnoDB 支持事务，MyISAM 不支持，这一点是非常之重要。事务是一种高级的处理方式，如在一些列增删改中只要哪个出错还可以回滚还原，而 MyISAM 就不可以了；

MyISAM 适合查询以及插入为主的应用，InnoDB 适合频繁修改以及涉及到安全性较高的应用；

InnoDB 支持外键，MyISAM 不支持；

MyISAM 是默认引擎，InnoDB 需要指定；

InnoDB 不支持 FULLTEXT 类型的索引；

InnoDB 中不保存表的行数，如 select count() from table 时，InnoDB；需要扫描一遍整个表来计算有多少行，但是 MyISAM 只要简单的读出保存好的行数即可。注意的是，当 count()语句包含 where 条件时 MyISAM 也需要扫描整个表；

对于自增长的字段，InnoDB 中必须包含只有该字段的索引，但是在 MyISAM 表中可以和其他字段一起建立联合索引；清空整个表时，InnoDB 是一行一行的删除，效率非常慢。MyISAM 则会重建表；

InnoDB 支持行锁（某些情况下还是锁整表，如 update table set a=1 where user like '%lee%'

## 10. 如何对查询命令进行优化？

a. 应尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索。

b. 应尽量避免在 where 子句中对字段进行 null 值判断，避免使用!=或<>操作符，避免使用 or 连接条件，或在 where 子句中使用参数、对字段进行表达式或函数操作，否则会导致权标扫描

c. 不要在 where 子句中的“=”左边进行函数、算术运算或其他表达式运算，否则系统将可能无法正确使用索引。

d. 使用索引字段作为条件时，如果该索引是复合索引，那么必须使用到该索引中的第一个字段作为条件时才能保证系统使用该索引，否则该索引将不会被使用。

e. 很多时候可考虑用 exists 代替 in。

f. 尽量使用数字型字段。

g. 尽可能的使用 varchar/nvarchar 代替 char/nchar。

h. 任何地方都不要使用 select from t ，用具体的字段列表代替“”，不要返回用不到的任何字段。

i. 尽量使用表变量来代替临时表。

j. 避免频繁创建和删除临时表，以减少系统表资源的消耗。

k. 尽量避免使用游标，因为游标的效率较差。

l. 在所有的存储过程和触发器的开始处设置 SET NOCOUNT ON ，在结束时设置 SETNOCOUNT OFF。

m. 尽量避免大事务操作，提高系统并发能力。

n. 尽量避免向客户端返回大数据量，若数据量过大，应该考虑相应需求是否合理。

## 11.数据库的优化？

1.优化索引、SQL 语句、分析慢查询；

2.设计表的时候严格根据数据库的设计范式来设计数据库；

3.使用缓存，把经常访问到的数据而且不需要经常变化的数据放在缓存中，能节约磁盘 IO

4.优化硬件；采用 SSD，使用磁盘队列技术(RAID0,RAID1,RDID5)等

5.采用 MySQL 内部自带的表分区技术，把数据分层不同的文件，能够提高磁盘的读取效率；

6.垂直分表；把一些不经常读的数据放在一张表里，节约磁盘 I/O；

7.主从分离读写；采用主从复制把数据库的读操作和写入操作分离开来；

8.分库分表分机器（数据量特别大），主要的原理就是数据路由；

9.选择合适的表引擎，参数上的优化

10.进行架构级别的缓存，静态化和分布式；

11.不采用全文索引；

12.采用更快的存储方式，例如 NoSQL 存储经常访问的数据。

## 12.Sql 注入是如何产生的，如何防止？

程序开发过程中不注意规范书写 sql 语句和对特殊字符进行过滤，导致客户端可以通过全局变量 POST 和 GET 提交一些 sql 语句正常执行。产生 Sql 注入。下面是防止办法：

a. 过滤掉一些常见的数据库操作关键字，或者通过系统函数来进行过滤。

b. 在 PHP 配置文件中将 Register_globals=off;设置为关闭状态

c. SQL 语句书写的时候尽量不要省略小引号(tab 键上面那个)和单引号

d. 提高数据库命名技巧，对于一些重要的字段根据程序的特点命名，取不易被猜到的

e. 对于常用的方法加以封装，避免直接暴漏 SQL 语句

f. 开启 PHP 安全模式：Safe_mode=on;

g. 打开 magic_quotes_gpc 来防止 SQL 注入

h. 控制错误信息：关闭错误提示信息，将错误信息写到系统日志。

i. 使用 mysqli 或 pdo 预处理。

## 13.NoSQL 和关系数据库的区别？

a. SQL 数据存在特定结构的表中；而 NoSQL 则更加灵活和可扩展，存储方式可以省是 JSON 文档、哈希表或者

其他方式。

b. 在 SQL 中，必须定义好表和字段结构后才能添加数据，例如定义表的主键(primary key)，索引(index),触发器(trigger),存储过程(stored procedure)等。表结构可以在被定义之后更新，但是如果有比较大的结构变更的话就会变得比较复杂。在 NoSQL 中，数据可以在任何时候任何地方添加，不需要

先定义表。

c. SQL 中如果需要增加外部关联数据的话，规范化做法是在原表中增加一个外键，关联外部数据表。而在 NoSQL 中除了这种规范化的外部数据表做法以外，我们还能用如下的非规范化方式把外部数据直接放到原数据集中，以提高查询效率。缺点也比较明显，更新审核人数据的时候将会比较麻烦。

d. SQL 中可以使用 JOIN 表链接方式将多个关系数据表中的数据用一条简单的查询语句查询出来。

NoSQL 暂未提供类似 JOIN 的查询方式对多个数据集中的数据做查询。所以大部分 NoSQL 使用非规范化的数据存储方式存储数据。

e. SQL 中不允许删除已经被使用的外部数据，而 NoSQL 中则没有这种强耦合的概念，可以随时删除任何数据。

f. SQL 中如果多张表数据需要同批次被更新，即如果其中一张表更新失败的话其他表也不能更新成

功。这种场景可以通过事务来控制，可以在所有命令完成后再统一提交事务。而 NoSQL 中没有事务这个概念，每一个数据集的操作都是原子级的。

g. 在相同水平的系统设计的前提下，因为 NoSQL 中省略了 JOIN 查询的消耗，故理论上性能上是优于 SQL 的。

## 14. MySQL 与 MongoDB 本质之间最基本的差别是什么

差别在多方面，例如：数据的表示、查询、关系、事务、模式的设计和定义、速度和性能。MongoDB 是由 C++ 语言编写的，是一个基于分布式文件存储的开源数据库系统。在高负载的情况下，添加更多的节点，可以保证服务器性能。

MongoDB 旨在为 WEB 应用提供可扩展的高性能数据存储解决方案。

MongoDB 将数据存储为一个文档，数据结构由键值(key=>value)对组成。MongoDB 文档类似于 JSON 对象。字段值可以包含其他文档，数组及文档数组。

MongoDB 是一个面向文档的数据库，目前由 10gen 开发并维护，它的功能丰富齐全，所以完全可以替代 MySQL。

与 MySQL 等关系型数据库相比，MongoDB 的优点如下：

①弱一致性，更能保证用户的访问速度。

②文档结构的存储方式，能够更便捷的获取数据。

③内置 GridFS，支持大容量的存储。

④内置 Sharding。

⑤第三方支持丰富。(这是与其他的 NoSQL 相比，MongoDB 也具有的优势)

⑥性能优越：

MongoDB 本身它还算比较年轻的一个产品，所以它的问题，就是成熟度肯定没有传统 MySQL 那么成熟稳定。

所以在使用的时候：

尽量使用稳定版，不要在线上使用开发版，这是一个大原则；

另外一点，备份很重要，MongoDB 如果出现一些异常情况，备份一定是要能跟上。除了通过传统的复制的方式来做备份，离线备份也还是要有，不管你是用什么方式，都要有一个完整的离线备份。往往最后出现了特殊情况，它能帮助到你；另外，MongoDB 性能的一个关键点就是索引，索引是不是能有比较好的使用效率，索引是不是能够放在内存中，这样能够提升随机读写的性能。如果你的索引不能完全放在内存中，一旦出现随机读写比较高的时候，它就会频繁地进行磁盘交换，这个时候，MongoDB 的性能就会急剧下降，会出现波动。

另外，MongoDB 还有一个最大的缺点，就是它占用的空间很大，因为它属于典型空间换时间原则的类型。那么它的磁盘空间比普通数据库会浪费一些，而且到目前为止它还没有实现在线压缩功能，在 MongoDB 中频繁的进行数据增删改时，如果记录变了，例如数据大小发生了变化，这时候容易产生一些数据碎片，出现碎片引发的结果，一个是索引会出现性能问题。

另外一个就是在一定的时间后，所占空间会莫名其妙地增大，所以要定期把数据库做修复，定期重新做索引，这样会提升 MongoDB 的稳定性和效率。在最新的版本里，它已经在实现在线压缩，估计应该在 2.0 版左右，应该能够实现在线压缩，可以在后台执行现在 repair DataBase 的一些操作。

如果那样，就解决了目前困扰我们的大问题。

## 15. Mysql 数据库中怎么实现分页？

select * from table limit (start-1)*limit,limit; 其中 start 是页码，limit 是每页显示的条数。

## 16. 提取数据库中倒数 10 条数据？

Select * from 表名 order by 主键 desc limit 10

## 17. Mysql 数据库的操作？

修改表-修改字段，重命名版：

alter table 表名 change 原名新名类型及约束；

alter table students change birthday birth datetime not null;

修改表-修改字段，不重名版本：

alter table 表名 modify 列名类型和约束；

alter table students modify birth date not null

全列插入：insert into 表名 values(...)

insert into students values(0,"郭靖", 1,"内蒙","2017-6");

部分插入：值的顺序与给出的列顺序对应：

insert into students(name, birthday) values("黄蓉","2017-8");

修改：update 表名 set 列 1=值 1，列 2=值 2.。。where

update students set gender=0, homwtown="古墓"， where id = 5;

备份：mysqldump -uroot -p 数据库名>python.sql,

恢复：mysql -uroot -p 数据库名< python.sql

## 18.优化数据库？提高数据库的性能？

1.对语句的优化

①用程序中，保证在实现功能的基础上，尽量减少对数据库的访问次数；通过搜索参数，尽量减少对表的访问行数,最小化结果集，从而减轻网络负担；

②能够分开的操作尽量分开处理，提高每次的响应速度；在数据窗口使用 SQL 时，尽量把使用的索引放在选择的首列；算法的结构尽量简单；

③在查询时，不要过多地使用通配符如 SELECT * FROM T1 语句，要用到几列就选择几列如：SELECT COL1,COL2 FROM T1；

④在可能的情况下尽量限制尽量结果集行数如：SELECT TOP 300 COL1,COL2,COL3 FROMT1,因为某些情况下用户是不需要那么多的数据的。

⑤不要在应用中使用数据库游标，游标是非常有用的工具，但比使用常规的、面向集的 SQL 语句需要更大的开销；按照特定顺序提取数据的查找。

2. 避免使用不兼容的数据类型

例如 float 和 int、char 和 varchar、binary 和 varbinary 是不兼容的。

数据类型的不兼容可能使优化器无法执行一些本来可以进行的优化操作。

例如:

SELECT name FROM employee WHERE salary ＞ 60000

在这条语句中,如 salary 字段是 money 型的,则优化器很难对其进行优化,因为 60000 是个整型数。我们应当在编程时将整型转化成为钱币型,而不要等到运行时转化。若在查询时强制转换，查询速度会明显减慢。

3.避免在 WHERE 子句中对字段进行函数或表达式操作。

若进行函数或表达式操作，将导致引擎放弃使用索引而进行全表扫描。

4.避免使用!=或＜＞、IS NULL 或 IS NOT NULL、IN ， NOT IN 等这样的操作符

5.尽量使用数字型字段

6.合理使用 EXISTS,NOT EXISTS 子句。

7.尽量避免在索引过的字符数据中，使用非打头字母搜索。

8.分利用连接条件

9.消除对大型表行数据的顺序存取

10. 避免困难的正规表达式

11. 使用视图加速查询

12. 能够用 BETWEEN 的就不要用 IN

13. DISTINCT 的就不用 GROUP BY

14. 部分利用索引

15. 能用 UNION ALL 就不要用 UNION

16. 不要写一些不做任何事的查询

17. 尽量不要用 SELECT INTO 语句

18. 必要时强制查询优化器使用某个索引

19. 虽然 UPDATE、DELETE 语句的写法基本固定，但是还是对 UPDATE 语句给点建议：

a) 尽量不要修改主键字段。

b) 当修改 VARCHAR 型字段时，尽量使用相同长度内容的值代替。

c) 尽量最小化对于含有 UPDATE 触发器的表的 UPDATE 操作。

d) 避免 UPDATE 将要复制到其他数据库的列。

e) 避免 UPDATE 建有很多索引的列。

f) 避免 UPDATE 在 WHERE 子句条件中的列。

## 19. 存储过程和函数的区别?

相同点：存储过程和函数都是为了可重复的执行操作数据库的 sql 语句的集合。

1）存储过程和函数都是一次编译，就会被缓存起来，下次使用就直接命中已经编译好的 sql 语句，不需要重复使用。减少网络交互，减少网络访问流量。

不同点：标识符不同，函数的标识符是 function，存储过程是 proceduce。

1）函数中有返回值，且必须有返回值，而过程没有返回值，但是可以通过设置参数类型（in,out)来实现多个参数或者返回值。

2）存储函数使用 select 调用，存储过程需要使用 call 调用。

3）select 语句可以在存储过程中调用，但是除了 select..into 之外的 select 语句都不能在函数中使用。

4）通过 in out 参数，过程相关函数更加灵活，可以返回多个结果。

## 20. Mysql 开启 General-log 日志?

Show variables like ‘general%’;

Set global general_log=1;

Set global general_log=0;

## 21. 请写出 truncate、 delete、 drop 的区别

1.删除的是部分行 delete from 表名....where

2.如果是清空表而且要保留主键最高值 delete from 表名

3.如果是清空表，而且主键也彻底删除 truncate table 表名

4.如果是彻底删除表，就有 drop table 表名

# 二、 查询练习

## 22. Student-Sourse-SC-Teacher 表关系如下：

- Student（sid，Sname，Sage，Ssex）学生表
- Course（cid，Cname，tid）课程表
- SC（sid，cid，score）成绩表
- Teacher（tid，Tname）教师表

写出 sql 语句：

- 查询课程“001“课程比”002“课程成绩高的所有学生的学号
- 修改学号为 20131201 的语文成绩为 100‘
- 插入一条名为“李四”的教师记录
- 删除学习“叶平”老师课程的 sc 表记录

## 23. 员工信息 A-员工亲属信息表 B 表关系如下：

- 员工信息表 A：员工标号（codecode，PK），员工姓名（codename），员工性别（codesex），联系电话（codetel），备注（remarks）
- 员工亲属信息表 B：员工编码（codecode），亲属编码（recodecode，PK），亲属姓名（recodename），联系电话（recodetel），备注（remarks）

写出 sql 语句：

- 向员工信息表中插入一条数据：（001，张三，男，010-62570007，北京市海淀区）
- 查询出亲属数量大于 1 的员工编码，员工姓名，员工亲属数量有部分员工亲属信息重复录入（亲属编码不同，其他相同），出现这种情况的员工编码，重复的亲属编码，亲属姓名查询出来。

## 24. 部门表 dept-雇员表 emp 表关系如下：

- 部门表 dept：部门标号（DEPTNO），部门名称（DNAME），所在位置（LOC）
- 雇员表 emp：员工标号（Empno），员工名称（Emname），员工工位（Job），经理（Mgr），雇佣日期（Hiredate），薪水（Sal），部门编号（Deptno）

写出 sql 语句：

- 找出部门名称为 ACCOUNTING 的部门下的所有员工名称？
- 找出部门名称为 SALES 的部门下每月需要发出的薪水总额？
- 找出部门名称为 SALES 的部门的部门经理？
- 找出部门名称为 RESEARCH 的部门下雇佣日期为 1980-12-17 的员工？

## 25. Student-coures-Studentcourse 表关系如下：

- student(sno,sname,age,sdept) 学生表
- course(cno,cname,teacher) 课程表
- Studentcourse(sno,cno,grade)选课表

写出 sql 语句：

- 查询所有课程都及格的学生号和姓名
- 查询平均分不及格的课程号和平均成绩
- 找出各门课程的平均成绩，输出课程号和平均成绩
- 找出没有选择 c2 课程的学生信息

## 26. 看下图回答问题

A1

| uid (用户id) | user (用户名) | post (帖子数) | hits (点击数) | data (日期) |
|---|---|---|---|---|
| 1 | 小张 | 90 | 983 | 2012-03-24 |
| 2 | 小王 | 123 | 243 | 2014-06-21 |
| 3 | 小三 | 998 | 100 | 2014-06-21 |
| 4 | 小三 | 782 | 983 | 2014-06-03 |
| 5 | 小三 | 344 | 334 | 2014-07-02 |
| ... | | | | |

1〉请用一条 SQL 语句查询出发帖数大于 5 点击数由高到低排序？

2〉用一条 SQL 语句将“日期”为 21 日的记录帖子数全部设置为 0？

3〉用一条 SQL 筛选出 2014 年的记录且 post 大约 500 的数据，将其 hits 减少 50？

## 27. SQL 操作，有两张表，如下图所示

订单表：A

| Order_id | User_id | Add_time |
|---|---|---|
| 11701245001 | 10000 | 1498882474 |
| 11701245002 | 10001 | 1498882475 |

订单明细表：B

| Id | Order_id | Goods_id | price |
|---|---|---|---|
| 1 | 11701245001 | 1001 | 10 |
| 2 | 11701245001 | 1002 | 20 |
| 3 | 11701245002 | 1001 | 10 |

问题 1:用 SQL 查询购买过 goods_id 为 1001 的用户的 user_id

问题 2:用 SQL 查询 2017 年 7 月 1 号后（含 7 月 1 号）购买过 1001 这个商品的 user_id 和 oeder_id, goods_id 和 price

问题 3:用 SQL 查询出订单所含商品明细总金额〉=50 的 order_id 和 user_id

## 28.题目

下面是学生成绩表（score）结构说明

| 字段名称 | 字段解释 | 字段类型 | 字段长度 | 约束 |
|---|---|---|---|---|
| sc_number | 学号 | 字符 | 8 | PK |
| sc_name | 姓名 | 字符 | 50 | Not null |
| sc_sex | 性别 | 字符(男：1；女：0) | 2 | Not null |
| sc_courseid | 课程号 | 字符 | 5 | PK |
| sc_score | 分数 | 数值 | 3 | Not null |
| sc_ismakeup | 当前考试是否为补考 | 字符(补考：1，非补考：0) | 2 | Not null |

下面是课程表（course）说明

| 字段名称 | 字段解释 | 字段类型 | 字段长度 | 约束 |
|---|---|---|---|---|
| co_id | 课程号 | 字符 | 5 | PK |
| co_name | 课程名 | 字符 | 3 | Not null |
| co_desc | 课程介绍 | 字符 | 60 | |

1） 请说明主键，外键的作用，以及建立索引的好处及坏处。

2）请写出课程表中建表 SQL 语句。

3）如果学号的前两位表示年级，要查找 98 级女生的姓名，请写出相应的 SQL 语句。

4）要查找所需要补考（小于 60 分）的学生姓名和这门课程的名称和成绩，请写出相应的 SQL 语句。

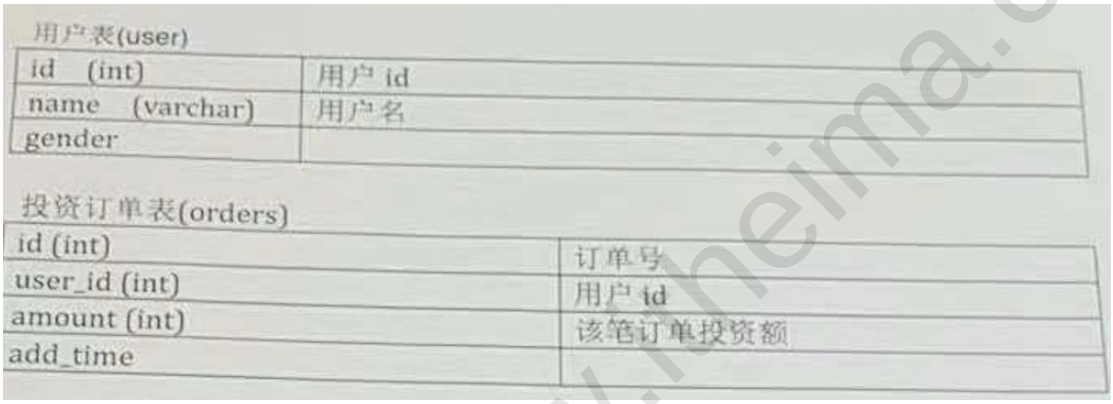
5） 查询每个学生需要补考（小于 60 分）的课程的平均分，并以平均分排序。

6）（非必答题，请量力选做）请分析这两个表的设计，并谈谈这两个表设计中存在的问题，并根据你的理

解给出优化的方案。

7）（非必答题，请量力选做）针对学生考试管理系统，如果要实现学生管理，课程管理，考试成绩管理的基本需求，请根据你对需求的理解，使用 UML 或者 E-R 图的方式给出一个简洁明了的数据库设计方案。

## 29.懒投资首页的懒人播报，统计了在懒投资平台的投资富豪榜，对应的库表简化如下：



用户表(user)

| | |
|---|---|
| id (int) | 用户 id |
| name (varchar) | 用户名 |
| gender | |

投资订单表(orders)

| | |
|---|---|
| id (int) | 订单号 |
| user_id (int) | 用户 id |
| amount (int) | 该笔订单投资额 |
| add_time | |

富豪榜统计需求：

1） 请给出，所有投资用户中，投资总额排名前 10 位的用户，按投资总额倒序排列，输出项如下，例如：

用户名，投资总额

张三，9000000

李四，8000000

……

2） 给出富豪榜第一名的用户的单笔平均投资额

# 第五章 Web 测试

## 1. 描述用浏览器访问 www.baidu.com 的过程？

先要解析出 baidu.com 对应的 ip 地址：

- 要先使用 arp 获取默认网关的 mac 地址
- 组织数据发送给默认网关(ip 还是 dns 服务器的 ip，但是 mac 地址是默认网关的 mac 地址)
- 默认网关拥有转发数据的能力，把数据转发给路由器
- 路由器根据自己的路由协议，来选择一个合适的较快的路径转发数据给目的网关
- 目的网关(dns 服务器所在的网关)，把数据转发给 dns 服务
- dns 服务器查询解析出 baidu.com 对应的 ip 地址，并原路返回请求这个域名的 client

得到了 baidu.com 对应的 ip 地址之后，会发送 tcp 的 3 次握手，进行连接

- 使用 http 协议发送请求数据给 web 服务器
- web 服务器收到数据请求之后，通过查询自己的服务器得到相应的结果，原路返回给浏览器
- 浏览器接收到数据之后通过浏览器自己的渲染功能来显示这个网页
- 浏览器关闭 tcp 连接，即 4 次挥手结束，完成整个访问过程

## 2. 了解的常用浏览器有哪些？

IE，Chrome，Safari，Firefox，Opera

## 3. 以京东首页为例，设计用例框架。（注意框架设计逻辑，区域划分，专项测试等，不需要详细用例，需要查看 PC 可直接和辨识管提要求）


## 4. 如何测试购买下单和退货流程

产品经理设计了单品优惠，组合优惠，订单优惠，优惠券优惠（优惠券优惠包含通用券，定向券，满减券，折扣券）和礼品卡，其中礼品卡上需要单独购买的。请问如何测试购买下单和退货流程，需要注意什么？（包含数据存储）

## 5. 什么是 sql 注入，什么是跨站脚本，什么是跨站请求伪造？

SQL 注入攻击是注入攻击最常见的形式（此外还有 OS 注入攻击（Struts 2 的高危漏洞就是通过 OGNL 实施 OS 注入攻击导致的)），当服务器使用请求参数构造 SQL 语句时，恶意的 SQL 被嵌入到 SQL 中交给数据库执行。SQL 注入攻击需要攻击者对数据库结构有所了解才能进行，攻击者想要获得表结构有多种方式：

（1）如果使用开源系统搭建网站，数据库结构也是公开的（目前有很多现成的系统可以直接搭建论坛，电商网站，虽然方便快捷但是风险是必须要认真评估的）；

（2）错误回显（如果将服务器的错误信息直接显示在页面上，攻击者可以通过非法参数引发页面错误从而通过错误信息了解数据库结构，Web 应用应当设置友好的错误页，一方面符合最小惊讶原则，一方面屏蔽掉可能给系统带来危险的错误回显信息）；

（3）盲注。防范 SQL 注入攻击也可以采用消毒的方式，通过正则表达式对请求参数进行验证，此外，参数绑定也是很好的手段，这样恶意的 SQL 会被当做 SQL 的参数而不是命令被执行，JDBC 中的 PreparedStatement 就是支持参数绑定的语句对象，从性能和安全性上都明显优于 Statement。

XSS（Cross Site Script，跨站脚本攻击）是向网页中注入恶意脚本在用户浏览网页时在用户浏览器中执行恶意脚本的攻击方式。跨站脚本攻击分有两种形式：

反射型攻击（诱使用户点击一个嵌入恶意脚本的链接以达到攻击的目标，目前有很多攻击者利用论坛、微博发布含有恶意脚本的 URL 就属于这种方式）

持久型攻击（将恶意脚本提交到被攻击网站的数据库中，用户浏览网页时，恶意脚本从数据库中被加载到页面执行，QQ 邮箱的早期版本就曾经被利用作为持久型跨站脚本攻击的平台）。

CSRF 攻击（Cross Site Request Forgery，跨站请求伪造）是攻击者通过跨站请求，以合法的用户身份进行非法操作（如转账或发帖等）。CSRF 的原理是利用浏览器的 Cookie 或服务器的 Session，盗取用户身份，其原理如下图所示。防范 CSRF 的主要手段是识别请求者的身份，主要有以下几种方式：

（1）在表单中添加令牌（token）；

（2）验证码；

（3）检查请求头中的 Referer（前面提到防图片盗链接也是用的这种方式）。

令牌和验证都具有一次消费性的特征，因此在原理上一致的，但是验证码是一种糟糕的用户体验，不是必要的情况下不要轻易使用验证码，目前很多网站的做法是如果在短时间内多次提交一个表单未获得成功后才要求提供验证码，这样会获得较好的用户体验。

## 6. 给你一个网站怎么开展测试？

a)首先，查找需求说明、网站设计等相关文档，分析测试需求。

b)制定测试计划，确定测试范围和测试策略，一般包括以下几个部分：功能性测试，界面测试，性能测试，数据库测试，安全性测试，.兼容性测试

c)设计测试用例：

- 功能性测试可以包括，但不限于以下几个方面：链接测试；链接是否正确跳转，是否存在空页面和无效页面，是否有不正确的出错信息返回等；提交功能的测试；多媒体元素是否可以正确加载和显示；多语言支持是否能够正确显示选择的语言等
- 界面测试可以包括但不限于以下几个方面：页面是否风格统一，美观。页面布局是否合理，重点内容和热点内容是否突出。控件是否正常使用。对于必须但为安装的空间，是否提供自动下载并安装的功能。文字检查。
- 性能测试一般从以下两个方面考虑：压力测试，负载测试，强度测试
- 数据库测试要具体决定是否需要开展。数据库一般需要考虑连结性，对数据的存取操作，数据内容的验证等方面。
- 安全性测试：基本的登录功能的检查；是否存在溢出错误，导致系统崩溃或者权限泄露；相关开发语言的常见安全性问题检查，例如 SQL 注入等；如果需要高级的安全性测试，确定获得专业安全公司的帮助，外包测试，或者获取支持。
- 兼容性测试，根据需求说明的内容，确定支持的平台组合：浏览器的兼容性；操作系统的兼容性；软件平台的兼容性；数据库的兼容性。

d)开展测试，并记录缺陷。合理的安排调整测试进度，提前获取测试所需的资源，建立管理体系（例如，需求变更、风险、配置、测试文档、缺陷报告、人力资源等内容）。

e)定期评审，对测试进行评估和总结，调整测试的内容。

## 7. 电商支付模块的测试如何展开？

支付流程里面就涉及到了第三方支付接口：

- 下单接口：商户提交下单请求到第三方支付接口，第三方支付收单成功后返回下单成功结果给到商户系统。（下单接口的最终处理结果分为下单成功和下单失败，若未收到明确结果可调用单笔订单查询接口查询结果。）
- 支付接口：调用该接口时指定支付参数，完成买家账户向商户账户的支付，采用页面跳转交互模式和后台通知交互模式。（结果分为两路返回：一路为前台在 return_url 页面跳转显示支付结果；一路为后台在 notify_url 收到支付结果通知后进行响应。）
- 退款接口：调用第三方支付的支付请求接口返回付款成功后，在需要做退款处理时调用退款请求接口发起退款处理。（退款接口的最终处理结果分为退款成功和退款失败，若未收到明确结果可调用退款查询接口查询结果。）

- 单笔订单查询接口：根据订单号查询单笔订单信息和状态。退款订单查询接口：调用第三方支付的退款接口返回后，在需要查询退款请求状态可调用退款订单查询接口查询退款订单的状态和订单信息。

测试过程中需要注意的主要测试点及异常场景：

- 首先要保证接口都能正常调用；
- 生成一笔订单，支付完成后，同步或异步重复回调，只有一次有效；
- 生成一笔订单，复制订单号和金额，再次生成一笔订单，用 fiddler 设置断点，用第一笔已完成的订单号和订单金额去替换现有的订单号和金额，无法完成支付；
- 生成一笔订单，跳转到第三方时修改金额，无法到账，或者如果是游戏充值游戏币的话，到账为篡改后的金额对应的游戏币；
- 异步通知屏蔽，同步有效，进行支付，同步能够正常到账；
- 同步设置无效，异步有效，进行支付，异步能够正常到账；
- 同步异步都设置无效，在第三方支付完成后，在重发机制时间范围内，设置异步有效，到下次通知时间点时，能够正常通知到账（补单机制的验证，如果商户收到第三方支付成功的通知后，要告知第三方支付收到了成功的通知，如果第三方支付收到商户应答不是 ok 或超时，第三方支付就会认为通知失败，会在规定的时间内持续调用 notify_url，一般有时间或次数的限制）；
- 针对支付订单在数据库中存储是否完整和正确进行校验（比如：第三方订单号--方便与第三方对账和问题排查、订单金额、订单状态等）；
- 如果是用户购买实物商品，用户发起退货，要保证退货流程正常，资金能正常返还，要考虑下并发情况的验证以确保安全性；
- 如果是用户购买虚拟商品，比如话费、油卡之类的商品，只有在发货失败的时候才能发起退货，注意验证；

## 8. 如何开展兼容性测试？

（1）Web 兼容性测试

- 首先开展人工测试，测试工程师测试主流浏览器和常用操作系统测试主流程和主界面，看看主流程和主界面是否有问题，如果存在问题，那么记录下 bug 情况，以及浏览器型号和版本，以及操作系统，准确定位 bug 产生的原因，提交 bug，告知开发人员修改。所有的主流设备都需要进行测试，只关注主流程和主界面，毕竟每个系统主流程和主界面不是很多，所以这个工作量还是可以承受的。
- 其次借助第三方测试工具，目前我觉得比较好用的第三方 Web 测试工具有 IEtester（离线）、SuperPreview（离线）和 Browsershots：browsershots.org（在线），一款可以测试 IE 的兼容，一款可以测试主流浏览器的兼容，包括谷歌、火狐、Opera 等等。借助第三方测试工具，找到 bug 产生的位置，分析测试结果，告知程序员调整。

（2）APP 兼容性测试

- APP 的兼容性测试和 Web 测试类似，首先开展人工测试，测试工程师借助测试设备对主流程和主功能，主界面进行测试；收集所有的能收集到的不同型号的测试设备测试主流程和主界面，看看主流程和主界面是否有问题，如果存在问题，综合考虑设备的使用率等因素，看看是否需要调整，如果需要，那么记录下 bug 情况以及测试设备的型号和操作系统，准确定位 bug 产生的原因，提交 bug，告知开发人员修改。
- 其次借助第三方测试工具，对于 APP 的兼容性测试，推荐的是百度众测平台和云测平台，我经常使用的是云测平台，这两款测试工具里面包含了安卓和 iOS 的测试；测试很齐全，包括功能测试、深度兼容测试、性能测试、网络环境测试，还可以模拟海量用户测试，，还可以导入自己编写的测试用例进行功能测试，里面还包括测试专家的测试，当然了找专家是要花钱滴。基本进行兼容性测试是不需要花钱的；测试工程师把打包好的 apk 或者 IPA 文件，上传到测试平台，选择需要测试的设备型号，开始任务即可；等待一段时间，在等待的时间你是不需要盯着的，你可以做其他的工作。测试完成后会生成一份测试报告，可以查看错误页面和错误日志，如果需要调整，那么提交 bug，告知程序员修改即可。

## 9. nginx,tomcat,apache 都是什么？

Nginx (engine x) 是一个高性能的 HTTP 和反向代理服务器，也是一个 IMAP/POP3/SMTP 服务器。

Apache HTTP Server 是一个模块化的服务器，源于 NCSAhttpd 服务器

Tomcat 服务器是一个免费的开放源代码的 Web 应用服务器，属于轻量级应用服务器，是开发和调试 JSP 程序的首选。

## 10. apache 和 nginx 的区别？

**Nginx 相对 Apache 的优点：**

轻量级，同样起 web 服务，比 apache 占用更少的内存及资源；

抗并发，nginx 处理请求是异步非阻塞的，支持更多的并发连接，而 apache 则是阻塞型的，在高并发下 nginx 能保持低资源低消耗高性能；

配置简洁；

高度模块化的设计，编写模块相对简单；

社区活跃。

**Apache 相对 Nginx 的优点：**

rewrite ，比 nginx 的 rewrite 强大；

模块超多，基本想到的都可以找到；

少 bug ，nginx 的 bug 相对较多；

超稳定。

## 11. Selenium 有哪些定位元素方法

1.id 定位

2.name 定位

3.class_name

4.tag_name

5.link_text

6.partial_link_text

7.xpath

8.css

当定位一个元素是，如果存在 ID 属性值时，我们可以优先考虑 ID 定位，当没有 ID，有 name 属性值和 class 属性值时也可以采用 name 和 class_name 定位。当次能确定该元素的标签为页面唯一时，可以采用 tag_name 定位，一般来说采用 link_text 定位的方法比较少。如果 ID ，name，class_name 单个定位不了，可以采用 css 和 xpath 定位。

# 第六章　API 测试

## 1. 接口类型有哪些？

接口是指外部系统与系统之间以及内部各子系统之间的交互点。

包括外部接口、内部接口，内部接口又包括：上层服务与下层服务接口、同级接口。

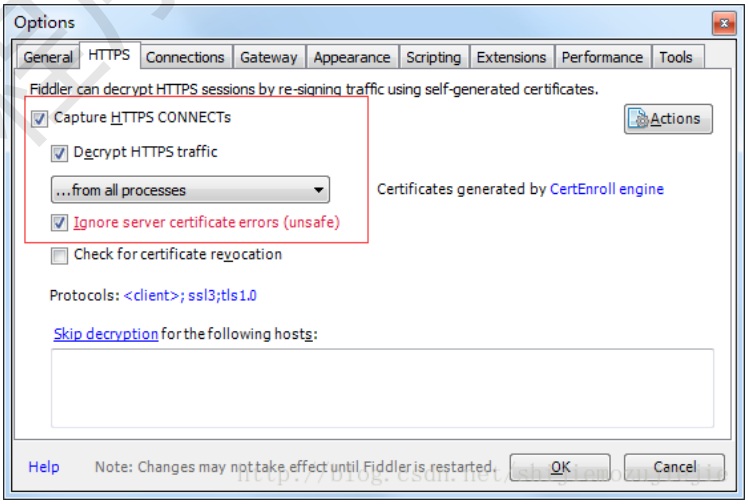
## 2. 如果模块请求 http 改为了 https，测试方案应该如何制定，修改？

分别用 http 还有 https 登录试试。如果用 https 可以正常登录，地址栏显示一把锁头，那么这个网站是有部署 SSL 的。如果 http 和 https 都能够正常登录，进一步说明该网站没有设置强制 https 登录，或者说没有设置 http 链接自动跳转 https 链接；相反如果用 http 登录，结果跳转到 https 页面，说明网站部署了 SSL，而且设置了 http 自动跳转 https。

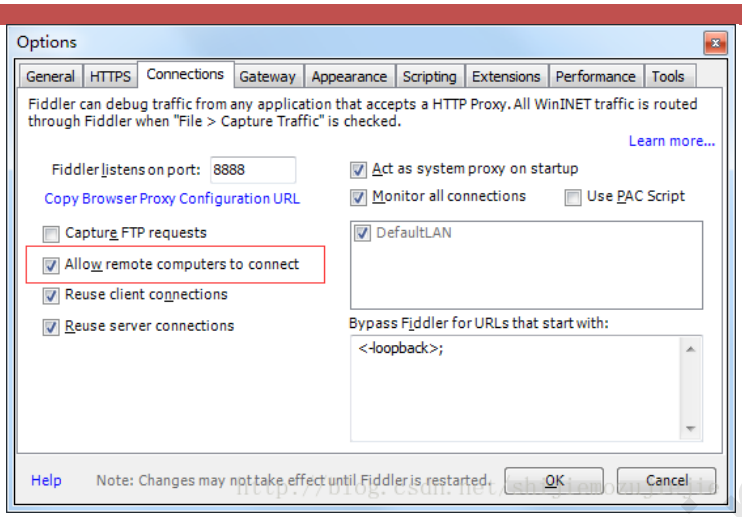
## 3. 常用 HTTP 协议调试代理工具有什么？详细说明抓取 HTTPS 协议的设置过程？

Fiddler 是一个 http 协议调试代理工具

打开 Fiddler，进入 Tools-Options-HTTPS，配置允许抓取 HTTPS 连接和解析 HTTPS 流量然后选择要解释的来源，设置是否忽略服务证书错误（这些操作做完之后，在浏览器方位 IP:8888，安装证书就可以在浏览器抓取 HTTPS 协议了）

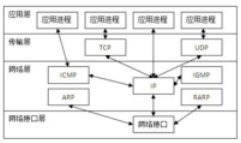



进入 Tools-Options-Connections，保证打开启抓取 HTTPS 连接，然后默认端口按需求是或否需要修改，然后点选允许远程计算机连接选项




## 4. 描述 TCP/IP 协议的层次结构，以及每一层中重要协议

一、TCP/IP网络体系结构中，常见的接口层协议有：Ethernet 802.3、Token Ring 802.5、X.25、Frame relay、HDLC、PPP ATM等。1.网络层网络层包括：IP(Internet Protocol ) 协议、ICMP(Internet Control Message Protocol) 、控制报文协议、ARP(Address Resolution Protocol ) 地址转换协议、RARP(Reverse ARP)反向地址转换协议。 2.传输层传输层协议主要是：传输控制协议TCP(Transmission Control Protocol ) 和用户数据报协议UDP(User Datagram protocol ) 。3.应用层应用层协议主要包括如下几个：FTP、TELNET、DNS、SMTP、RIP、NFS、HTTP。

、

## 5. jmeter，一个接口的响应结果如下：

<!DOCTYPE html>

<html>

<head>

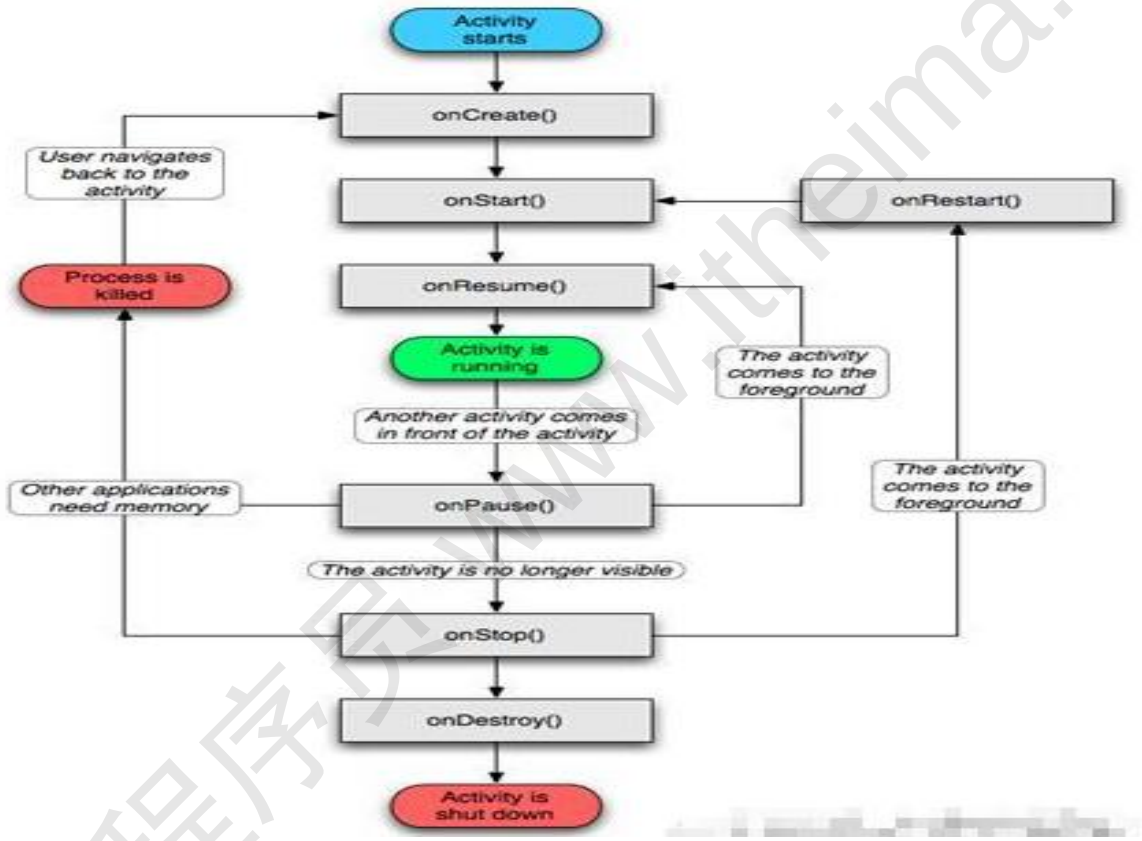
<title newBidId = “74956”encryptBidId=”713504275825”>小贷公司标管理</title>

请用正则表达式方法分别获取一下 74956 和 713504275825 这两个数值分别赋值给 A1 和 A2

# 第七章 App 测试

## 1. 简述 Android 四大组件及生命周期？

Android 的四大组件包括：Activity、Service、BroadcasReceiver、ContentProvider




## 2. 当点击 APP 图标启动程序，说明将要发生那些过程？

1.点击桌面 app 图标，Launcher 进程采用 Binder IPC 向 system_server 进程发起 startActivity 请求；

2.system_server 进程收到请求后，向 zygote 进程发送创建进程的请求（zygote 进程是 Android 系统的第一个进程，zygote 意为受精卵，所有进程都是由它孵化而来）

3.zygote 进程 fork 出新的子进程，即 App 进程；

4.App 进程，通过 Binder IPC 向 system_server 进程发起 attachApplication 请求

5.system_server 进程收到请求后，进行一系列的准备工作，通过 Binder IPC 向 App 进程发送 scheduleLaunchActivity 请求

6.App 的 binder 线程（Application Thread）在收到请求后，通过 handler 向主线程发送 LAUNCH_ACTIVITY 消息

7.主线程收到 Message 后，通过发射机制创建目标 Activity，并回调 Activity.onCreate()方法

## 3. APP 测试的内容主要包括哪些，如何开展？

功能测试：

1.业务逻辑正确性测试：依据：产品文档->测试用例编写

兼容性测试：

1.系统版本：Android:官方版本,定制版本;IOS：官方提供版本

2.分辨率：720 * 1280 1080* 1920

3.网络情况:2g 3g 4g 5g Wi-Fi

异常测试

1.热启动应用:应用在后台长时间待机;应用在后台待机过程中，手机重启

2.网络切换和中断恢复:网络切换;中断恢复：

3.电话信息中断恢复

升级，安装，卸载测试

1.升级测试：临近版本升级(1.0->1.1);跨版本(1.0->....->2.2)

2.安装测试：首次安装;覆盖安装(同版本，不同版本覆盖);卸载后安装

3.卸载测试：首次卸载;卸载安装后在卸载

健壮性测试

1.手机资源消耗：cpu，内存

2.流量消耗：图片，数据，视频

3.电量测试

4.崩溃恢复

## 4. Android 的兼容性测试都考虑哪些内容？

品牌机型兼容：根据市场占有率、发布时间等指标对主流、最新机型进行重点兼容

ROM 兼容：需兼容原生的 ROM（2.1、2.2、2.3、4.0、4.1、4.2）；第三方 ROM（小米、百度易、点心、魅族、阿里云……）

屏幕兼容：需兼容 HVGA、VGA、WVGA、FWVGA、720p、1080p 屏幕分辨率，并考虑不同 PPI 的情况

软件兼容：安全类软件（百度手机管家、360 优化大师、360 安全卫士、QQ 手机管家、安卓优化大师、网秦、LBE），输入法软件（系统自带、Sogou、百度）

版本兼容：服务器端需要兼容产品早期版本所需的 API 接口

网络兼容：WiFi、3 大运营商的 2G,3G,4G 网络，需区分 WAP 和 NET 接入

## 5. 针对 App 的安装功能，写出测试点？

- 安装

1.正常安装测试，检查是否安装成功。

2.APP 版本覆盖测试。例如：先安装一个 1.0 版本的 APP,再安装一个高版本(1.1 版本)的 APP，检查是否被覆盖。

3.回退版本测试。例如：先装一个 2.0 版本的 APP,再安装一个 1.0 版本的 APP,正常情况下版本是可以回退的。

4.安装时内存不足，弹出提示。

5.根据安装手册操作，是否正确安装。

6.安装过程中的意外情况（强行断电、断网、来电话了、查看信息）等等，检查会发生的情况。

7.通过'同步软件'，检查安装时是否同步安装了一些文件。

8.在不同型号、系统、屏幕大小、分辨率上的手机进行安装。

9.安装时是否识别有 SD 卡，并默认安装到 sd 卡中。

10.安装完成后，能否正常启动应用程序。

11.安装完成后，重启手机能否正常启动应用程序。

12.安装完成后，是否对其他应用程序造成影响。

13.安装完成后，能否添加快捷方式。

14.安装完成后，杀毒软件是否会对其当做病毒处理。

15.多进程进行安装，是否安装成功。

16.在安装过程中，所有的提示信息必须是英文或者中文，提示信息中不能出现代码、符号、乱码等。

17.安装之后，是否自动启动程序。

18.是否支持第三方安装。

19.在安装中点击取消。

- 卸载

1.用自己的卸载程序进行卸载，检查是否卸载干净。

2.用第三方工具，检查是否卸载干净。

3.在卸载过程中，点击取消按钮，看是否正常退出卸载程序，检查软件是否还能继续正常使用。

4.卸载过程中，出现意外（比如手机关机，没电，查看信息，接打电话），程序是否还能运行。

5.在卸载过程中，突然重启设备，再次访问程序，是否还能运行。

6.在没用使用程序时，删除目录文件，看程序是否能运行。

7.在使用过程中，直接删除目录文件，程序是否还能运行。

8.不同系统、硬件环境、网络环境下进行卸载。

9.卸载成功后，是否对其他程序有影响。

10.卸载后再次安装，是否正常使用。

11.在卸载过程中，所有的提示信息必须是英文或者中文，提示信息中不能出现代码、符号、乱码等。

- 更新

1.当客户端有新版本时，提示更新。

2.非强制更新，可以取消更新，旧版本正常使用，下次使用软件时，仍然会出现更新提示。

3.强制更新，强制更新而用户没有更新时，退出客户端，下次启动，依然提示更新。

4.不卸载更新，检查是否可以更新。

5.不卸载更新，检查资源同名文件如图片等是否更新成最新版本。

6.非 wifi 网络下，提示是否更新，取消就加入待下载，wifi 下自动更新。

## 6. 常用的 ADB 命令？

adb --help / adb :看见帮助信息

adb start-server:启动 adb 服务

adb kill-server:关闭 adb 服务

adb devices:查看手机设备号

adb shell getprop ro.build.version.release:获取系统版本

adb push 电脑 手机

adb pull 手机 电脑

adb logcat | grep(unix) 包名

adb logcat | findstr(win) 包名

adb shell :进入 shell 命令行，可以操作 Linux 命令

adb shell dumpsys window windows | grep mFocusedApp:获取包名 启动名(win：adb shell dumpsys window windows | findstr mFocusedApp)

adb install 路径/apk 文件:安装 apk 到手机上

adb uninstall 包名:卸载 app 从手机上

adb shell am start -W 包名/启动名:app 启动时间

## 7. adb 命令你知道哪些？adb shell dumpsys dumpsys 是什么意思？做什么的？

## 8. 在查看 logcat 命令日志时候怎么内容保存到本地文件？

输出重定向：logcat >> log_file_name

## 9. App 崩溃（闪退），可能是什么原因导致的？

缓存垃圾过多：由于安卓系统的特性,如果长时间不清理垃圾文件.会导致越来越卡.也会出现闪退情况.

运行的程序过多,导致内存不足

应用版本兼容问题：如果应用版本太低，会导致不兼容，造成闪退。此外，有些新版本在调试中，也会造成应用闪退。解决方法：如果是版本太旧，更新为新版本即可；如果是新版本闪退，可能是应用在改版调试，可卸载后安装旧版。

检查 APP 中访问网络的地方，组件中的 ImageView 是否可以正常的下载并显示到 app 页面上。

检查 APP 的 sdk 和手机的系统是否兼容。

在一些特定情况下的闪退,比如播放视频,在 Android5.0 升级到 Android6.0 的时候,有些系统 API 老版本有,新版本没有,到时回去对象的时候失败,报空,系统就会出现闪退问题.

## 10. 如何测试监测 app 的内存使用、CPU 消耗、流量使用情况？

adb shell top

Android 应用性能测试通常包括：启动时间、内存、CPU、耗电量、流量、流畅度等

根据手机的使用应用频度和强度不同，可将应用使用强度分为如下几种状态：

- 空闲状态：指启动应用后，不做任何操作或切换到后台运行的情况称为空闲状态，该情况为应用对内存的消耗是最小的。
- 中强度状态：该情况用户使用应用的强度和时间长短不确定，相对来说使用时长偏长。
- 高强度状态：该种情况为应用内高频率的使用，用户很少达到，跑 monkey 时可认为高强度状态，该种情况常用来测试应用内存泄漏的情况测试时，可根据用户的操作习惯模拟应用使用频率和强度等级。

使用 adb 命令，手机连接电脑开启 USB 调试模式，进入 adbshell。

（1）查看 CPU 占用率

使用命令 top -m 10 -s cpu（-t 显示进程名称，-s 按指定行排序，-n 在退出前刷新几次，-d 刷新间隔，-m 显示最大数量），如下图：

```
User 31%, System 25%, IOW 0%, IRQ 0%
User 262 + Nice 14 + Sys 225 + Idle 382 + IOW 5 + IRQ 0 + SIRQ 2 = 890

  PID PR CPU% S  #THR     VSS     RSS PCY UID      Name
30522  2  32% R    89 1240904K 115244K  fg u0_a177  com.baidu.tieba
  255  0   7% R    19 149396K   4256K  fg system   /system/bin/surfacefling
  788  1   2% S   113 1457960K 115224K  fg system   system_server
10054  0   1% R     1   4520K   1500K  fg shell    top
25607  0   1% S   128 1276172K  77504K  bg u0_a127  com.tencent.mobileqq
  290  5   0% S     5  25052K    560K  fg system   /system/bin/aal
10099  2   0% S     1      0K      0K  fg root     kworker/u16:5
 8442  2   0% S     1      0K      0K  fg root     kworker/u16:2
23515  0   0% S    56 1344228K  41224K  bg u0_a130  com.mfashiongallery.ema
 7535  1   0% S     1      0K      0K  fg root     kworker/u16:7
```

参数含义：

- PID：progressidentification，应用程序 ID

- S: 进程的状态，其中 S 表示休眠，R 表示正在运行，Z 表示僵死状态，N 表示该进程优先值是负数。
- #THR：程序当前所用的线程数
- VSS：VirtualSet Size 虚拟耗用内存（包含共享库占用的内存）
- RSS： ResidentSet Size 实际使用物理内存（包含共享库占用的内存）
- UID：UserIdentification，用户身份 ID
- Name:应用程序名称

在测试过程中，QA 需要关注对应包的 cpu 占用率，反复进行某个操作，cpu 占用过高且一直无法释放，此时可能存在风险。如果你想筛选出你自己的应用的话可以用下面命令 top -d 3| grep packageName

```
130|shell@hermes:/ $ top -d 3| grep com.baidu.tieba
30522  1   0% S    79 1190024K  93004K  fg u0_a177  com.baidu.tieba
30595  1   0% S    29 1018196K  43296K  fg u0_a177  com.baidu.tieba:remote
30522  2  16% R    79 1190024K  93456K  fg u0_a177  com.baidu.tieba
30595  2   0% S    29 1018196K  43296K  fg u0_a177  com.baidu.tieba:remote
30522  2  18% S    79 1190352K  93744K  fg u0_a177  com.baidu.tieba
30595  2   0% S    29 1018196K  43296K  fg u0_a177  com.baidu.tieba:remote
30522  0  18% R    79 1190352K  93908K  fg u0_a177  com.baidu.tieba
30595  2   0% S    29 1018196K  43296K  fg u0_a177  com.baidu.tieba:remote
30522  0  20% S    79 1190352K  94260K  fg u0_a177  com.baidu.tieba
30595  0   0% S    29 1018196K  43296K  fg u0_a177  com.baidu.tieba:remote
30522  1   8% S    79 1190352K  94260K  fg u0_a177  com.baidu.tieba
30595  2   0% S    29 1018196K  43296K  fg u0_a177  com.baidu.tieba:remote
30522  0   0% S    79 1190352K  94260K  fg u0_a177  com.baidu.tieba
30595  2   0% S    29 1018196K  43296K  fg u0_a177  com.baidu.tieba:remote
```

（2）查看内存使用情况

dumpsys meminfo <package_name>或 dumpsys meminfo <package_id>

```
255|shell@hermes:/ $ dumpsys meminfo 30522
Applications Memory Usage (kB):
Uptime: 31122761 Realtime: 160602067

** MEMINFO in pid 30522 [com.baidu.tieba] **
                   Pss  Private  Private  Swapped     Heap     Heap     Heap
                 Total    Dirty    Clean    Dirty     Size    Alloc     Free
                ------   ------   ------   ------   ------   ------   ------
  Native Heap     9908     9856        0     1896    18572    12792     1507
  Dalvik Heap    29494    29236        0    16132    46560    45625      935
 Dalvik Other     2260     2188        0        0
        Stack      700      700        0        0
       Ashmem      128      124        0        0
    Other dev        5        0        4        0
     .so mmap     5395      432     3872     1680
    .apk mmap      250        0      160        0
    .ttf mmap      919        0      836        0
    .dex mmap     6680        0     6032        0
   code mmap     3712        0     1224        0
  image mmap     3714     1752      152      192
  Other mmap      434        8      392        0
     Graphics    12488    12488        0        0
           GL    14419    14419        0        0
      Unknown     7628     7600        0       56
        TOTAL    98134    78803    12672    19956    65132    58417     2442

 Objects
               Views:      233         ViewRootImpl:        1
         AppContexts:        9           Activities:        1
              Assets:        7        AssetManagers:        7
       Local Binders:       38        Proxy Binders:       36
    Death Recipients:        3
     OpenSSL Sockets:        0
```

参数含义：

- Naitve Heap Size: 从 mallinfo usmblks 获得，代表最大总共分配空间
- Native Heap Alloc: 从 mallinfo uorblks 获得，总共分配空间
- Native Heap Free: 从 mallinfo fordblks 获得，代表总共剩余空间
- Native Heap Size 约等于 Native Heap Alloc + Native Heap Free
- mallinfo 是一个 C 库， mallinfo 函数提供了各种各样的通过 C 的 malloc（）函数分配的内存的统计信息。
- Dalvik Heap Size:从 Runtime totalMemory()获得，Dalvik Heap 总共的内存大小。
- Dalvik Heap Alloc: Runtime totalMemory()-freeMemory() ，Dalvik Heap 分配的内存大小。
- Dalvik Heap Free:从 Runtime freeMemory()获得，Dalvik Heap 剩余的内存大小。
- Dalvik Heap Size 约等于 Dalvik HeapAlloc + Dalvik Heap Free

重点关注如下几个字段：

- Native/Dalvik 的 Heap 信息中的 alloc ：具体在上面的第一行和第二行，它分别给出的是 JNI 层和 Java 层的内存分配情况，如果发现这个值一直增长，则代表程序可能出现了内存泄漏。

Total 的 PSS 信息：这个值就是你的应用真正占据的内存大小，通过这个信息，你可以轻松判别手机中哪些程

序占内存比较大了。

### 11.弱网测试怎么测

弱网环境测试主要依赖于弱网环境的模拟。环境搭建方式一般有两种：软件方式和硬件方式。软件方式的成本低，主要就是通过模拟网络参数来配置弱网环境，通常来讲可以达到测试目的.一般可通过热点共享设置。在各类网络软件中，主要就是对带宽、丢包、延时等进行模拟弱网环境。如果要求更接近弱网环境，比如现在很多的专项测试，会更倾向于通过硬件方式来协助测试，但这种方式相对会麻烦很多，一般会由网维协助搭建。当然，对于有些无法模拟的情况，只能靠人工移动到例如电梯、地铁等信号比较弱的地方。

### 12.如何定位 APP 上的元素，使用 appium 的 inspector 了吗，用什么平台做的

### 13.如何使用 xpath 定位一个兄弟元素

### 14. “//*[contains(@text,”登录”)]”是什么意思

定位第一个元素 text 属性包含登录的元素

### 15.自己最熟悉的哪个库，如何使用这些库的，是否做了基于复用的封装，怎么考虑的这些封装

### 16.自动化测试的框架画一遍，然后解释

### 17.Monkey 命令

### 18.微信客户端使用搜狗输入法打字，手机屏幕突然黑了，请问有哪些原因会导致这个现象，分别如何进行排查

### 19.微信 APP 有什么地方需要改进的地方

### 20.如何选择测试手机，如何购买测试手机，多少台测试手机

### 21.如何选择功能做自动化测试

### 22.自动化测试占比

### 23.如何用 fiddler 或者 Charles 进行测试手机

### 24.Appium 都有哪些启动方式

1.客户端启动

2.命令行启动

**25.** 请写出你所了解的手机操作系统并简述各自的特点

**26. Android ROM** 是否了解 不同 **ROM** 有什么区别 **target SDK** 是什么

**27. Xpath** 定位不到的情况

**28.** 移动端的注册登录和网页端的有什么区别？

**29.** 怎么搭移动端自动化框架的，需要哪些内容？

**30. APP** 在运行过程中为什么会出现卡顿？

**31.** 请写一下支付宝首页的测试用例（要求：逻辑清晰）

# 第八章　管理工具

1. **熟悉的软件项目管理工具有哪些？**
2. **结合你的测试工作中使用的管理缺陷的工具，讲一下使用此工具描述软件缺陷和跟踪管理流程?**
3. **简述常用的 Bug 管理或者用例管理工具,并且描述其中一个工作流程？**

常用：testlink，QC，mantis，禅道，TAPD，JIRA 。

TAPD：产品创建(需求，计划，模块)-->项目创建（PM 排期、任务分解）-->研发(编码、单元 测试等)-->测试(测试计划，用例，执行，bug，报告等)。

4. **禅道和 qc 的区别？**

同为缺陷管理工具。

QC

作为缺陷管理工具，QC 在缺陷管理方面，做的相对完善。提 bug 页面：填写内容可以根据测试需求，不断修改添加新的字段；以我上一家公司为例，在提 bug 过程中，有一下几个必填项：Bug 状态（new、fixed、closed 等）、发现人员、缺陷发现阶段(测试阶段、上现阶段等)、缺陷来源（测试人员给出的 bug 定位）、Bug 分类（功能、性能等问题）、测试阶段（单元测试、集成测试、系统测试等）、归属需求、缺陷回归次数、优先级、分配给，这些必填项再加上 bug 标题和操作描述、上传附件，使很多疑问都变得清晰。

缺陷查看页面：可以根据自己需要选择要呈现的字段，相对人性化可操作，每个显示的字段都可以进行筛选，使研发人员很快能定位到属于自己的 bug，再根据 bug 状态、优先级进行筛选，使未完结的 bug 能有序并无遗漏地完成修改；页面还有注释功能，研发人员能写出针对本问题的各种感想，方便完善而又人性化。

禅道（开源版）

禅道涉及面非常广，但是在缺陷管理这方面，与老牌的 QC 还是略逊一筹。提 bug 页面：页面是非常清晰整洁的 web 页面，但是需要填写的字段，并没有完全覆盖开发和测试人员的全部需求。页面字段：产品模块（对应 QC 中的项目）、所属项目（对应 QC 中的需求）、影响版本（bug 所属版本？）、当前指派（修改 bug 的人员）、bug 标题、重现步骤、相关需求（页面标注了这个字段，但是什么也没有显示，并且没有可填写的位置）、相关任务、类型/严重。

# 第九章　Python 基础

## 1. 斐波那契数列求 N?

```
N=int(input("N:"))
fib=[0 for x in range(N+1)]
fib[0]=0
fib[1]=1
for i in range(2,N+1):
    fib[i]=fib[i-1]+fib[i-2]
print fib[N]
```

## 2. 字符串反序输出？

print(a_str[::-1])

## 3. 判断回文？

astr[::-1] == a_str

## 4. 统计 python 源代码文件中代码行数，去除注释，空行，进行输出？

```
#-*-coding:utf-8-*-

import os

if __name__=='__main__':
    codeline=0
    expline=0
    blankline=0

    filename=raw_input('Please input the  file name:')

    fi=open(filename)
    while fi.tell()!=os.path.getsize(filename):
        temp=fi.readline()
        if temp.startswith('#'):
            expline+=1
        elif temp=='/n':
            blankline+=1
        elif temp.startswith('"""'):
            expline+=1
            while True:
                temp=fi.readline()
                expline+=1
                if temp.endswith('"""/n'):
                    break
        else:
            codeline+=1
    else:
        print 'the codeline is:'+str(codeline)
        print 'the expline is:'+str(expline)
        print 'the blank line is:'+str(blankline)
```

## 5. python 调用 cmd 并返回结果？

python 的 OS 模块。

- OS 模块调用 CMD 命令有两种方式：os.popen(),os.system(). 都是用当前进程来调用。

- os.system 是无法获取返回值的。当运行结束后接着往下面执行程序。用法如：OS.system("ipconfig").
- OS.popen 带返回值的，如何获取返回值。如
- p=os.popen(cmd)
- print p.read().得到的是个字符串。
- 这两个都是用当前进程来调用，也就是说它们都是阻塞式的。

管道 subprocess 模块。

- 运行原理会在当前进程下面产生子进程。
- sub=subprocess.Popen(cmd,shell=True,stdout=subprocess.PIPE)
- sub.wait()
- print sub.read()

## 6. 冒泡排序

```
def bsort(alist):
    l = len(alist)
    for i in range(l-1):
        flag = 1
        for j in range(l-i-1):
            if alist[j] > alist[j+1]:
                alist[j],alist[j+1] = (alist[j+1],alist[j])
                flag = 0
        if flag == 1:
            return alist
```

## 7. 1,2,3,4 这 4 个数字，能组成多少个互不相同的且无重复的三位数，都是多少？

```
i = 0
for x in range(1,5):
    for y in range(1,5):
        for z in range(1,5):
                if (x!=y) and (y!=z) and (z!=x):
                    i += 1
                    if i%4:
                        print("%d%d%d" % (x, y, z), end=" | ")
                    else:
                        print("%d%d%d" % (x, y, z))
```

## 8. 给定一个整数 N，和一个 0-9 的数 K，要求返回 0-N 中数字 K 出现的次数

```
def digitCounts(self, k, n):
            count = 0
        for i in range(n+1):
```

```
        if i == 0 and i == k:
            count   += 1
        while( i // 10 >= 0 and i != 0):
            j = i % 10
            if j == k:
                count   += 1
            i = i //10
    return count
```

**9. 请用 python 打印出 10000 以内的对称数（对称数特点：数字左右对称，如：1,2,11,121,1221 等）**

**10. 判断 101-200 之间有多少个素数，并输出所有的素数**

```
from math import sqrt
#定义一个是否素数函数，如果n等于1，则返回false
def is_prime(n):
    if n == 1:
        return False
    for i in range(2,int(sqrt(n))+1):
        if n%i == 0:
            return False
    return True
```

**11. 一个输入三角形的函数，输入后输出是否能组成三角形，三角形类型，请用等价类划分法设计测试用例**

```
a,b,c=map(int ,input().split())
if a<+c and b<a+c and c<a+b:
    if a==b==c:
        print('等边三角形')
    elif a==b or a==c or b==c:
        if a*a+b*b==c*c or a*a+c*c==b*b or b*b+c*c==a*a:
            print('等腰直角三角形')
            else:
            print('等腰三角形')
```

```
elif a*a+b*b==c*c or a*a+c*c==b*b or b*b+c*c==a*a:
        print('直角三角形')
    else:
        print('普通三角形')
    else:
        print('无法构成三角形')
```

**12.给出一组[ ‘-’，‘-’，‘+’，‘+’，’ +’，’ -’，’ +’，’ -’，’ +’，’ -’，’ -’ ]符号，使用你最熟悉的语言，把“+”号放在 list 的左边，“-”号放在 list 的右边。**

**13.编程题**

1）请编写一个完整出程序，实现如下功能：从键盘输入数字 n，程序自动计算 n!，并输出。（注 1:n!=1*2*3*……*n，注 2:请使用递归实现）（可以使用任何开发语言，最好使用 JAVA）

2） 如果现在有一台刚安装了 WinXP 的计算机，请简单说明如何能够让以上程序得以运行。

3. 写代码将如下数据从小到大排序，语言不限。（不可以直接使用 sort（）等排序方法）

234，82，5，10，86，90

4.如何使用 Python 发送一封邮件？

5. Linux 下如何查看 ip 地址，如何用 Python 或 TCL 删除当前文件夹下所有文件以及目录？

6.给 x 变量赋值为 abccaefs，并统计 x 变量中单词出现的次数（java 或 Python 任选一种语言编写）

# 一、 输入与输出

**14.代码中要修改不可变数据会出现什么问题？抛出什么异常？**

代码不会正常运行，抛出 TypeError 异常。

**15.代码中要修改不可变数据会出现什么问题？抛出什么异常？**

代码不会正常运行，抛出 TypeError 异常。

**16.print 调用 Python 中底层的什么方法？**

print 方法默认调用 sys.stdout.write 方法，即往控制台打印字符串。

### 17. 简述你对 input()函数的理解?

在 Python3 中，input()获取用户输入，不论用户输入的是什么，获取到的都是字符串类型的。

在 Python2 中有 raw_input()和 input(), raw_input()和 Python3 中的 input()作用是一样的，input()输入的是什么数据类型的，获取到的就是什么数据类型的。

### 18. python 两层列表怎么提取第二层的元素

aa = [[(55736,)], [(55739,)], [(55740,), (55801,)], [(55748,)], [(55783,), (55786,), (55787,), (55788,)], [(55817,), (55821,)], [(55818,)]]

```
def getelement(aa):
   for elem in aa:
      if type(elem)==type([]):
         for element in getelement(elem):
            yield element
      else:yield elem
for elem in getelement(aa):print(elem)(55736,)
```

(55739,)

(55740,)

(55801,)

(55748,)

(55783,)

(55786,)

(55787,)

(55788,)

(55817,)

(55821,)

(55818,)

## 二、 条件与循环

### 19. 阅读下面的代码，写出 A0，A1 至 An 的最终值？

```
1.  A0 = dict(zip(('a'，'b'，'c'，'d'，'e')，(1，2，3，4，5)))
2.  A1 = range(10)
3.  A2 = [i for i in A1 if i in A0]
```

```
4.  A3 = [A0[s] for s in A0]
5.  A4 = [i for i in A1 if i in A3]
6.  A5 = {i:i*i for i in A1}
7.  A6 = [[i，i*i] for i in A1]
```

答

```
1.  A0 = {'a': 1， 'c': 3， 'b': 2， 'e': 5， 'd': 4}
2.  A1 = [0， 1， 2， 3， 4， 5， 6， 7， 8， 9]
3.  A2 = []
4.  A3 = [1， 3， 2， 5， 4]
1.  A4 = [1， 2， 3， 4， 5]
2.  A5 = {0: 0， 1: 1， 2: 4， 3: 9， 4: 16， 5: 25， 6: 36， 7: 49， 8: 64， 9: 81}
3.  A6 = [[0， 0]， [1， 1]， [2， 4]， [3， 9]， [4， 16]， [5， 25]， [6， 36]，[7， 49]，[8， 64] [9，81]]
```

### 20. range 和 xrange 的区别？

两者用法相同，不同的是 range 返回的结果是一个列表，而 xrange 的结果是一个生成器，前者是直接开辟一块内存空间来保存列表，后者是边循环边使用，只有使用时才会开辟内存空间，所以当列表很长时，使用 xrange 性能要比 range 好。

### 21. 考虑以下 Python 代码，如果运行结束，命令行中的运行结果是什么？

```
1.  l = []
2.  for iin xrange(10):
3.      l.append({‘num’:i})
4.  print l
```

在考虑以下代码，运行结束后的结果是什么？

```
1.  l = []
2.  a = {‘num’:0}
3.  for i in xrange(10):
4.      a[‘num’] = i
5.  l.append(a)
6.  print l
```

## 三、 字典

**dict:**字典，字典是一组键(key)和值(value)的组合，通过键(key)进行查找，没有顺序， 使用大括号”{}”；

应用场景：

**dict**，使用键和值进行关联的数据;

### 22. 现有字典 d={‘a’:24，‘g’:52，‘i’:12，‘k’:33}请按字典中的 value 值进行排序？

sorted(d.items()， key = lambda x:x[1]) 。

### 23. 说一下字典和 json 的区别？

字典是一种数据结构，json 是一种数据的表现形式，字典的 key 值只要是能 hash 的就行，json 的必须是字符串。

### 24. 什么是可变、不可变类型？

可变不可变指的是内存中的值是否可以被改变，不可变类型指的是对象所在内存块里面的值不可以改变，有数值、字符串、元组；可变类型则是可以改变，主要有列表、字典。

### 25. 存入字典里的数据有没有先后排序？

存入的数据不会自动排序，可以使用 sort 函数对字典进行排序。

### 26. 字典推导式？

d = {key: value for (key, value) in iterable}

### 27. 现有字典 d={‘a’:24，’g’:52，’l’:12，’k’:33}请按字 典中的 value 值进行排序？

sorted(d.items()， key = lambda x:x[1])

## 四、 字符串

str:字符串是 Python 中最常用的数据类型。我们可以使用引号('或")来创建字符串。

### 28. 如何理解 Python 中字符串中的\字符？

有三种不同的含义：

1、转义字符 2、路径名中用来连接路径名 3、编写太长代码手动软换行。

**29. 请反转字符串“aStr”?**

print(‘aStr’[::-1])

### 30.请按 alist 中元素的 age 由大到小排序

alist [{'name':'a'，'age':20}，{'name':'b'，'age':30}，{'name':'c'，'age':25}]

def sort_by_age(list1):

return sorted(alist，key=lambda x:x['age']，reverse=True)

## 五、 列表

list:是 Python 中使用最频繁的数据类型，在其他语言中通常叫做数组，通过索引进行查找，使用方括号”[]”,列表是有序的集合。应用场景：定义列表使用 [ ] 定义，数据之间使用 “，”分割。

列表的索引从 0 开始：索引就是数据在列表中的位置编号，索引又可以被称为下标。

【注意】：从列表中取值时,如果超出索引范围,程序会产生异常。

IndexError: list index out of range

### 31.列表增加

列表名.insert(index, 数据)：在指定位置插入数据(位置前有空元素会补位)。

# 往列表 name_list 下标为 0 的地方插入数据

In [3]: name_list.insert(0, "Sasuke")

In [4]: name_list

Out[4]: ['Sasuke', 'zhangsan', 'lisi', 'wangwu', 'zhaoliu']

# 现有的列表下标是 0-4，如果我们要在下标是 6 的地方插入数据，那个会自动插入到下标为 5 的地方，也就是# 插入到最后

In [5]: name_list.insert(6, "Tom")

In [6]: name_list

Out[6]: ['Sasuke', 'zhangsan', 'lisi', 'wangwu', 'zhaoliu', 'Tom']

列表名.append(数据)：在列表的末尾追加数据(最常用的方法)。

In [7]: name_list.append("Python")

In [8]: name_list

Out[8]: ['Sasuke', 'zhangsan', 'lisi', 'wangwu', 'zhaoliu', 'Tom', 'Python']

列表.extend(Iterable)：将可迭代对象中的元素追加到列表。

# 有两个列表 a 和 b a.extend(b) 会将 b 的元素追加到列表

In [10]: a = [11, 22, 33]

```
In [11]: b = [44, 55, 66]
In [12]: a.extend(b)
In [13]: a
Out[13]: [11, 22, 33, 44, 55, 66]# 有列表 c 和 字符串 c
In [14]: c = ['j', 'a', 'v', 'a']
In [15]: d = "python"
In [16]: c.extend(d)
In [17]: c
Out[17]: ['j', 'a', 'v', 'a', 'p', 'y', 't', 'h', 'o', 'n']
```

## 32.取值和修改取值：列表名[index] ：根据下标来取值。

```
In [19]: name_list = ["zhangsan", "lisi", "wangwu", "zhaoliu"]
In [20]: name_list[0]
Out[20]: 'zhangsan'
In [21]: name_list[3]
Out[21]: 'zhaoliu'
```

修改：列表名[index] = 数据：修改指定索引的数据。

```
In [22]: name_list[0] = "Sasuke"
In [23]: name_list
Out[23]: ['Sasuke', 'lisi', 'wangwu', 'zhaoliu']
```

## 33.删除 del 列表名[index]：删除指定索引的数据。

```
In [25]: name_list = ["zhangsan", "lisi", "wangwu", "zhaoliu"]# 删除索引是 1 的数据
In [26]: del name_list[1]
In [27]: name_list
Out[27]: ['zhangsan', 'wangwu', 'zhaoliu']
```

## 34.列表名.remove(数据)：删除第一个出现的指定数据。

```
In [30]: name_list = ["zhangsan", "lisi", "wangwu", "zhaoliu", "lisi"]# 删除 第一次出现的 lisi 的数据
In [31]: name_list.remove("lisi")
In [32]: name_list
Out[32]: ['zhangsan', 'wangwu', 'zhaoliu', 'lisi']
```

## 35.列表名.pop()：删除末尾的数据,返回值：返回被删除的元素。

```
In [33]: name_list = ["zhangsan", "lisi", "wangwu", "zhaoliu"]# 删除最后一个元素
In [34]: name_list.pop()
Out[34]: 'zhaoliu'
In [35]: name_list
Out[35]: ['zhangsan', 'lisi', 'wangwu']
```

## 36.列表名.pop(index)：删除指定索引的数据，返回被删除的元素。

```
In [36]: name_list = ["zhangsan", "lisi", "wangwu", "zhaoliu"]# 删除索引为 1 的数据 lisi
In [37]: name_list.pop(1)
Out[37]: 'lisi'
In [38]: name_list
Out[38]: ['zhangsan', 'wangwu', 'zhaoliu']
```

## 37.列表名.clear()：清空整个列表的元素。

```
In [40]: name_list = ["zhangsan", "lisi", "wangwu", "zhaoliu"]
In [41]: name_list.clear()
In [42]: name_list
Out[42]: []
```

## 38.排序列表名.sort()：升序排序 从小到大。

```
In [43]: a = [33, 44, 22, 66, 11]
In [44]: a.sort()
In [45]: a
Out[45]: [11, 22, 33, 44, 66]
```

## 39.列表名.sort(reverse=True)：降序排序 从大到小。

```
In [46]: a = [33, 44, 22, 66, 11]
In [47]: a.sort(reverse=True)
In [48]: a
Out[48]: [66, 44, 33, 22, 11]
```

## 40.列表名.reverse()：列表逆序、反转。

```
In [50]: a = [11, 22, 33, 44, 55]
In [51]: a.reverse()
In [52]: a
Out[52]: [55, 44, 33, 22, 11]
```

## 41.len(列表名)：得到列表的长度。

```
In [53]: a = [11, 22, 33, 44, 55]
In [54]: len(a)
Out[54]: 5
```

## 42.列表名.count(数据)：数据在列表中出现的次数。

```
In [56]: a = [11, 22, 11, 33, 11]
In [57]: a.count(11)
Out[57]: 3
```

## 43.列表名.index(数据)：数据在列表中首次出现时的索引，没有查到会报错。

```
In [59]: a = [11, 22, 33, 44, 22]
In [60]: a.index(22)
Out[60]: 1
```

## 44.if 数据 in 列表： 判断列表中是否包含某元素。

```
a = [11, 22, 33, 44 ,55]
if 33 in a:
    print("找到了....")
```

## 45.循环遍历

使用 while 循环：

```
a = [11, 22, 33, 44, 55]
    i = 0
while i < len(a):
    print(a[i])
```

```
    i += 1
```

使用 for 循环：

```
a = [11, 22, 33, 44, 55]
for i in a:
    print(i)
```

## 46.写一个列表生成式，产生一个公差为 11 的等差数列

```
print([x*11 for x in range(10)])
```

## 47.给定两个列表，怎么找出他们相同的元素和不同的元素?

```
list1 = [1，2，3]
list2 = [3，4，5]
set1 = set(list1)
set2 = set(list2)
print(set1&set2)
print(set1^set2)
```

## 48.请写出一段 Python 代码实现删除一个 list 里面的重复元素?

比较容易记忆的是用内置的 set：

```
l1 = ['b'，'c'，'d'，'b'，'c'，'a'，'a']
l2 = list(set(l1))
print l2
```

如果想要保持他们原来的排序：用 list 类的 sort 方法：

```
l1 = ['b'，'c'，'d'，'b'，'c'，'a'，'a']
l2 = list(set(l1))
```

```
3    l2.sort(key=l1.index)
4    print l2
```

也可以这样写：

```
1    l1 = ['b'，'c'，'d'，'b'，'c'，'a'，'a']
2    l2 = sorted(set(l1)，key=l1.index)
3    print l2
```

也可以用遍历：

```
1    l1 = ['b', 'c', 'd', 'b', 'c', 'a', 'a']
2    l2 = []
3    for i in l1:
4        if not i in l2:
5            l2.append(i)
6    print l2
```

### 49.给定两个 list A ,B，请用找出 A ,B 中相同的元素，A ,B 中不同的元素

A、B 中相同元素：print(set(A)&set(B))

A、B 中不同元素：print(set(A)^set(B))

新的默认列表只在函数被定义的那一刻创建一次。当 extendList 被没有指定特定参数 list 调用时，这组 list 的值随后将被使用。这是因为带有默认参数的表达式在函数被定义的时候被计算，不是在调用的时候被计算。

## 六、 元组

tuple:元组，元组将多样的对象集合到一起，不能修改，通过索引进行查找，使用括号”()”；应用场景：把一些数据当做一个整体去使用，不能修改

# 七、 集合

set:set 集合，在 Python 中的书写方式的{}，集合与之前列表、元组类似，可以存储多个数据，但是这些数据是不重复的。集合对象还支持 union(联合), intersection(交), difference(差)和 ysmmetric_difference(对称差集)等数学运算.

## 50.快速去除列表中的重复元素

```
In [4]: a = [11,22,33,33,44,22,55]
In [5]: set(a)
Out[5]: {11, 22, 33, 44, 55}
```

## 51.交集：共有的部分

```
In [7]: a = {11,22,33,44,55}
In [8]: b = {22,44,55,66,77}
In [9]: a&b
Out[9]: {22, 44, 55}
```

## 52.并集：总共的部分

```
In [11]: a = {11,22,33,44,55}
In [12]: b = {22,44,55,66,77}
In [13]: a | b
Out[13]: {11, 22, 33, 44, 55, 66, 77}
```

## 53.差集：另一个集合中没有的部分

```
In [15]: a = {11,22,33,44,55}
In [16]: b = {22,44,55,66,77}
In [17]: b - a
Out[17]: {66, 77}
```

## 54.对称差集(在 a 或 b 中，但不会同时出现在二者中)

```
In [19]: a = {11,22,33,44,55}
In [20]: b = {22,44,55,66,77}
In [21]: a ^ b
Out[21]: {11, 33, 66, 77}
```

# 八、 文件操作

## 55.4G 内存怎么读取一个 5G 的数据？(2018-3-30-lxy)

方法一：

可以通过生成器，分多次读取，每次读取数量相对少的数据（比如 500MB）进行处理，处理结束后在读取后面的 500MB 的数据。

方法二：

可以通过 linux 命令 split 切割成小文件，然后再对数据进行处理，此方法效率比较高。可以按照行数切割，可以按照文件大小切割。

## 56.现在要处理一个大小为 10G 的文件，但是内存只有 4G，如果在只修改 get_lines 函数而其他代码保持不变的情况下，应该如何实现？需要考虑的问题都有哪些？

```
1. def get_lines():
2. l = []
3. with open('file.txt', 'rb') as f:
4.     data = f.readlines(60000)
5.     l.append(data)
6.     yield l
```

要考虑到的问题有：内存只有 4G 无法一次性读入 10G 的文件，需要分批读入。分批读入数据要记录每次读入数据的位置。分批每次读入数据的大小，太小就会在读取操作上花费过多时间。

## 57.read、readline 和 readlines 的区别?

read:读取整个文件。 readline：读取下一行，使用生成器方法。

readlines：读取整个文件到一个迭代器以供我们遍历。

# 九、 函数

## 58.Python 函数调用的时候参数的传递方式是值传递还是引用传递？

Python 的参数传递有：位置参数、默认参数、可变参数、关键字参数。

函数的传值到底是值传递还是引用传递，要分情况：

不可变参数用值传递：像整数和字符串这样的不可变对象，是通过拷贝进行传递的，因为你无论如何都不可能在原处改变

不可变对象

可变参数是引用传递的：

比如像列表，字典这样的对象是通过引用传递、和 C 语言里面的用指针传递数组很相似，可变对象能在函数内部改变。

### 59.对缺省参数的理解 ？

缺省参数指在调用函数的时候没有传入参数的情况下，调用默认的参数，在调用函数的同时赋值时，所传入的参数会替代默认参数。

*args 是不定长参数，他可以表示输入参数是不确定的，可以是任意多个。

**kwargs 是关键字参数，赋值的时候是以键 = 值的方式，参数是可以任意多对在定义函数的时候不确定会有多少参数会传入时，就可以使用两个参数。

### 60.为什么函数名字可以当做参数用?

Python 中一切皆对象，函数名是函数在内存中的空间，也是一个对象。

### 61.Python 中 pass 语句的作用是什么？

在编写代码时只写框架思路，具体实现还未编写就可以用 pass 进行占位，使程序不报错，不会进行任何操作。

## 十、 内建函数

### 62.map 函数和 reduce 函数？

①从参数方面来讲：

map()包含两个参数，第一个参数是一个函数，第二个是序列（列表 或元组）。其中，函数（即 map 的第一个参数位置的函数）可以接收一个或多个参数。

reduce()第一个参数是函数，第二个是序列（列表或元组）。但是，其函数必须接收两个参数。

②从对传进去的数值作用来讲：

map()是将传入的函数依次作用到序列的每个元素，每个元素都是独自被函数“作用”一次 。

reduce()是将传人的函数作用在序列的第一个元素得到结果后，把这个结果继续与下一个元素作用（累积计算）。

### 63.递归函数停止的条件？

递归的终止条件一般定义在递归函数内部，在递归调用前要做一个条件判断，根据判断的结果选择是继续调用自身，还是 return;返回终止递归。

终止的条件：

判断递归的次数是否达到某一限定值

判断运算的结果是否达到某个范围等，根据设计的目的来选择

### 64.回调函数，如何通信的?

回调函数是把函数的指针(地址)作为参数传递给另一个函数，将整个函数当作一个对象，赋值给调用的函数。

### 65.Python 主要的内置数据类型都有哪些？ print dir( ‘a ’) 的输出？

内建类型：布尔类型、数字、字符串、列表、元组、字典、集合；输出字符串‘a’的内建方法；

### 66.print(list(map(lambda x: x * x, [y for y in range(3)])))的输出？

[0， 1， 4]

## 十一、 Lambda

### 67.什么是 lambda 函数？ 有什么好处？

lambda 函数是一个可以接收任意多个参数(包括可选参数)并且返回单个表达式值的函数

lambda 函数比较轻便，即用即仍，很适合需要完成一项功能，但是此功能只在此一处使用，连名字都很随意的情况下；

匿名函数，一般用来给 filter， map 这样的函数式编程服务;

作为回调函数，传递给某些应用，比如消息处理

### 68.什么是 lambda 函数？它有什么好处？写一个匿名函数求两个数的和？

lambda 函数是匿名函数；使用 lambda 函数能创建小型匿名函数。这种函数得名于省略了用 def 声明函数的标准步骤；

```
1        f = lambda x，y:x+y
```

```
.
    2           print(f(2017，2018))
.
```

# 十二、 面向对象

## 69. Python 中的可变对象和不可变对象？

不可变对象，该对象所指向的内存中的值不能被改变。当改变某个变量时候，由于其所指的值不能被改变，相当于把原来的值复制一份后再改变，这会开辟一个新的地址，变量再指向这个新的地址。

可变对象，该对象所指向的内存中的值可以被改变。变量（准确的说是引用）改变后，实际上是其所指的值直接发生改变，并没有发生复制行为，也没有开辟新的出地址，通俗点说就是原地改变。

Python 中，数值类型（int 和 float）、字符串 str、元组 tuple 都是不可变类型。而列表 list、字典 dict、集合 set 是可变类型。

## 70. Python 中 is 和==的区别？

is 判断的是 a 对象是否就是 b 对象，是通过 id 来判断的。

==判断的是 a 对象的值是否和 b 对象的值相等，是通过 value 来判断的。

## 71. Python 的魔法方法？

魔法方法就是可以给你的类增加魔力的特殊方法，如果你的对象实现 （重载）了这些方法中的某一个，那么这个

方法就会在特殊的情况下被 Python 所调用，你可以定义自己想要的行为，而这一切都是自动发生的。 它们经常是两个下划线包围来命名的（比如 __init__，__lt__），Python 的魔法方法是非常强大的，所以了解其使用方法也变得尤为重要！__init__ 构造器，当一个实例被创建的时候初始化的方法。但是它并 不是实例化调用的第一个方法。

__new__才是实例化对象调用的第一个方法，它只取下 cls 参数，并把 其他参数传给 __init__。 __new__很少使用，但是也有它适合的场景，尤其 是当类继承自一个像元组或者字符串这样不经常改变的类型的时候。

__call__ 允许一个类的实例像函数一样被调用 。

__getitem__ 定义获取容器中指定元素的行为，相当于 self[key] 。

__getattr__ 定义当用户试图访问一个不存在属性的时候的行为 。

__setattr__ 定义当一个属性被设置的时候的行为 。

__getattribute__ 定义当一个属性被访问的时候的行为 。

## 72.面向对象中怎么实现只读属性?

1、将对象私有化，通过共有方法提供一个读取数据的接口。

```
class person:
    def __init__(self,x):
        self.__age = 10;
    def age(self):
        return self.__age;
t = person(22)
# t.__age = 100
print(t.age())
```

2、最好的方法

```
class MyCls(object):
    __weight = 50
    @property #以访问属性的方式来访问  weight  方法
    def weight(self):
        return self.__weight
if __name__ == '__main__':
    obj = MyCls()
    print(obj.weight)
    obj.weight = 12
```

```
Traceback (most recent call last):
50
File "C:/PythonTest/test.py", line 11, in <module>
    obj.weight = 12
AttributeError: can't set attribute
```

## 73.谈谈你对面向对象的理解？

面向对象是相对于面向过程而言的。面向过程语言是一种基于功能分析的、以算法为中心的程序设计方法；而面向对象是一种基于结构分析的、以数据为中心的程序设计思想。在面向对象语言中有一个有很重要东西，叫做类。面向对象有三大特性：封装、继承、多态。

# 十三、 正则表达式

## 74. Python 里 match 与 search 的区别？

match()函数只检测 RE 是不是在 string 的开始位置匹配， search()会扫描整个 string 查找匹配；也就是说 match()只有在 0 位置匹配成功的话才有返回，如果不是开始位置匹配成功的话，match()就返回 none。

## 75. Python 字符串查找和替换？

re.findall(r’目的字符串’，’原有字符串’) #查询

re.findall(r'cast'，'itcast.cn')[0]

re.sub(r‘要替换原字符’，’要替换新字符’，’原始字符串’)

re.sub(r'cast'，'heima'，'itcast.cn')

## 76. 用 Python 匹配 HTML g tag 的时候，<.*> 和 <.*?> 有什么区别？

<.*>是贪婪匹配，会从第一个“<”开始匹配，直到最后一个“>”中间所有的字符都会匹配到，中间可能会包含“<>”。

<.*?>是非贪婪匹配，从第一个“<”开始往后，遇到第一个“>”结束匹配，这中间的字符串都会匹配到，但是不会有“<>”。

## 77. 请写出下列正则关键字的含义？

| 语法 | 说明 | 表达式实例 | 完整匹配的字符串 |
|---|---|---|---|
| 字符 | | | |
| 一般字符 | 匹配自身 | abc | abc |

| | | | |
|---|---|---|---|
| . | 匹配任意除换行符"\n"外的字符。<br>在 DOTALL 模式中也能匹配换行符。 | a.c | abc |
| \ | 转义字符,使后一个字符改变原来的意思。<br>如果字符串中有字符*需要匹配,可以使用*或者字符集[*] | a\.c a\\c | a. c a\c |
| [...] | 字符集(字符类)。对应的位置可以是字符集中任意字符。字符集中的字符可以逐个列出,也可以给出范围,<br>如[abc]或[a-c]。第一个字符如果是^ 则表示取反,如[^abc]表示不是 abc 的其他字符。<br>所有的特殊字符在字符集中都失去其原有的特殊含义。在字符集中如果要使用]、-或^,可以在前面加上反斜杠,或把]、-放在第一个字符,把^ 放在非第—个字符。 | a[bcd]e | abe ace ade |
| 预定义字符集(可以写在字符集[…]中) | | | |
| \d | 数字:[0-9] | a\dc | a1c |
| \D | 非数字:[^\d] | a\Dc | abc |
| \s | 空白字符:[<空格>\ t\r\n\f\v] | a\sc | ac |
| \S | 非空白字符:[^\s] | a\Sc | abc |
| \w | 单词字符:[A-Za-z0-9_] | a\wc | abc |
| \W | 非单词字符:[^\W] | a\Wc | ac |
| | 数量词(用在字符或(…)之后) | | |
| * | 匹配前一个字符 0 或无限次。 | abc* | ab abccc |
| + | 匹配前一个字符 1 次或无限次 | abc+ | abc abccc |
| ? | 匹配前一个字符 0 次或 1 次。 | abc? | Ab abc |
| {m} | 匹配前一个字符 m 次 | ab{2}c | abbc |

# 十四、 异常

**78.在 except 中 return 后还会不会执行 finally 中的代码？怎么抛出自定义**

异常？

会继续处理 finally 中的代码；用 raise 方法可以抛出自定义异常。

### 79.介绍一下 except 的作用和用法？

except: #捕获所有异常 except: <异常名>: #捕获指定异常

except:<异常名 1, 异常名 2>: 捕获异常 1 或者异常 2

except:<异常名>,<数据>:捕获指定异常及其附加的数据

except:<异常名 1,异常名 2>:<数据>:捕获异常名 1 或者异常名 2,及附加的数据

## 十五、 模块与包

### 80.常用的 Python 标准库都有哪些？

os 操作系统，time 时间，random 随机，pymysql 连接数据库，threading 线程，multiprocessing 进程，queue 队列。第三方库：

django 和 flask 也是第三方库，requests，virtualenv，selenium，scrapy，xadmin，celery，re，hashlib，md5。常用的科学计算库（如 Numpy，Scipy，Pandas)。

### 81.赋值、浅拷贝和深拷贝的区别？

一、 赋值在 Python 中，对象的赋值就是简单的对象引用，这点和 C++不同，如下所示：

```
a = [1,2,"hello",['python', 'C++']]
b = a
```

在上述情况下，a 和 b 是一样的，他们指向同一片内存，b 不过是 a 的别名，是引用。

我们可以使用 b is a 去判断，返回 True，表明他们地址相同，内容相同，也可以使用 id()函数来查看两个列表的地址是否相同。

赋值操作(包括对象作为参数、返回值)不会开辟新的内存空间，它只是复制了对象的引用。也就是说除了 b 这个名字之外，没有其他的内存开销。修改了 a，也就影响了 b，同理，修改了 b，也就影响了 a。

## 二、 浅拷贝(shallow copy)

浅拷贝会创建新对象，其内容非原对象本身的引用，而是原对象内第一层对象的引用。

浅拷贝有三种形式:切片操作、工厂函数、copy 模块中的 copy 函数。

比如上述的列表 a；

切片操作：b = a[:] 或者 b = [x for x in a]；工厂函数：b = list(a)；

copy 函数：b = copy.copy(a)；

浅拷贝产生的列表 b 不再是列表 a 了，使用 is 判断可以发现他们不是同一个对象，使用 id 查看，他们也不指向同一片内存空间。但是当我们使用 id(x) for x in a 和 id(x) for x in b 来查看 a 和 b 中元素的地址时，可以看到二者包含的元素的地址是相同的。

在这种情况下，列表 a 和 b 是不同的对象，修改列表 b 理论上不会影响到列表 a。

但是要注意的是，浅拷贝之所以称之为浅拷贝，是它仅仅只拷贝了一层，在列表 a 中有一个嵌套的 list，如果我们修改了它，情况就不一样了。

比如：a[3].append('java')。查看列表 b，会发现列表 b 也发生了变化，这是因为，我们修改了嵌套的 list，修改外层元素，会修改它的引用，让它们指向别的位置，修改嵌套列表中的元素，列表的地址并未发生变化，指向的都是用一个位置。

## 三、深拷贝(deep copy)

深拷贝只有一种形式，copy 模块中的 deepcopy()函数。

深拷贝和浅拷贝对应，深拷贝拷贝了对象的所有元素，包括多层嵌套的元素。因此，它的时间和空间开销要高。

同样的对列表 a，如果使用 b = copy.deepcopy(a)，再修改列表 b 将不会影响到列表 a，即使嵌套的列表具有更深的层次，也不会产生任何影响，因为深拷贝拷贝出来的对象根本就是一个全新的对象，不再与原来的对象有任何的关联。

## 四、拷贝的注意点？

对于非容器类型，如数字、字符，以及其他的“原子”类型，没有拷贝一说，产生的都是原对象的引用。

如果元组变量值包含原子类型对象，即使采用了深拷贝，也只能得到浅拷贝。

## 82.__init__ 和__new__的区别？

init 在对象创建后，对对象进行初始化。

new 是在对象创建之前创建一个对象，并将该对象返回给 init。

## 83.Python 里面如何生成随机数？

在 Python 中用于生成随机数的模块是 random，在使用前需要 import. 如下例子可以酌情列举：

random.random()：生成一个 0-1 之间的随机浮点数； random.uniform(a, b)：生成[a,b]之间的浮点数；random.randint(a, b)：生成[a,b]之间的整数；

random.randrange(a, b, step)：在指定的集合[a,b)中，以 step 为基数随机取一个数；random.choice(sequence)：从特定序列中随机取一个元素，这里的序列可以是字符串，列表，元组等。

## 84.输入某年某月某日，判断这一天是这一年的第几天？(可以用 Python 标准库)

```
import datetime
def dayofyear():
year = input("请输入年份：")
month = input("请输入月份：")
day = input("请输入天：")
date1 = datetime.date(year=int(year)，month=int(month)，day=int(day))
date2 = datetime.date(year=int(year)，month=1，day=1)
return (date1 - date2 + 1).days
```

## 85.打乱一个排好序的 list 对象 alist？

```
import random
random.shuffle(alist)
```

## 86.说明一下 os.path 和 sys.path 分别代表什么？

os.path 主要是用于对系统路径文件的操作。 sys.path 主要是对 Python 解释器的系统环境参数的操作（动态的改变 Python 解释器搜索路径）。

## 87.Python 中的 os 模块常见方法？

- os.remove()删除文件
- os.rename()重命名文件
- os.walk()生成目录树下的所有文件名
- os.chdir()改变目录
- os.mkdir/makedirs 创建目录/多层目录
- os.rmdir/removedirs 删除目录/多层目录
- os.listdir()列出指定目录的文件
- os.getcwd()取得当前工作目录
- os.chmod()改变目录权限
- os.path.basename()去掉目录路径，返回文件名
- os.path.dirname()去掉文件名，返回目录路径
- os.path.join()将分离的各部分组合成一个路径名
- os.path.split()返回（dirname(),basename())元组
- os.path.splitext()(返回 filename,extension)元组
- os.path.getatime\ctime\mtime 分别返回最近访问、创建、修改时间
- os.path.getsize()返回文件大小 os.path.exists()是否存在
- os.path.isabs()是否为绝对路径
- os.path.isdir()是否为目录
- os.path.isfile()是否为文件

## 88.Python 的 sys 模块常用方法？

- sys.argv 命令行参数 List，第一个元素是程序本身路径
- sys.modules.keys() 返回所有已经导入的模块列表
- sys.exc_info() 获取当前正在处理的异常类,exc_type、exc_value、exc_traceback 当前处理的
- 异常详细信息
- sys.exit(n) 退出程序，正常退出时 exit(0)
- sys.hexversion 获取 Python 解释程序的版本值，16 进制格式如：0x020403F0
- sys.version 获取 Python 解释程序的版本信息
- sys.maxint 最大的 Int 值
- sys.maxunicode 最大的 Unicode 值
- sys.modules 返回系统导入的模块字段，key 是模块名，value 是模块
- sys.path 返回模块的搜索路径，初始化时使用 PYTHONPATH 环境变量的值
- sys.platform 返回操作系统平台名称
- sys.stdout 标准输出
- sys.stdin 标准输入
- sys.stderr 错误输出
- sys.exc_clear() 用来清除当前线程所出现的当前的或最近的错误信息
- sys.exec_prefix 返回平台独立的 python 文件安装的位置
- sys.byteorder 本地字节规则的指示器，big-endian 平台的值是'big',little-endian 平台的值是
- 'little'
- sys.copyright 记录 python 版权相关的东西
- sys.api_version 解释器的 C 的 API 版本
- sys.version_info 元组则提供一个更简单的方法来使你的程序具备 Python 版本要求功能

## 89.模块和包是什么

在 Python 中，模块是搭建程序的一种方式。每一个 Python 代码文件都是一个模块，并可以引用其他的模块，比如对象和属性。

一个包含许多 Python 代码的文件夹是一个包。一个包可以包含模块和子文件夹。

# 十六、 Python 特性

## 90. Python 是强语言类型还是弱语言类型？

Python 是强类型的动态脚本语言。

强类型：不允许不同类型相加。

动态：不使用显示数据类型声明，且确定一个变量的类型是在第一次给它赋值的时候。

脚本语言：一般也是解释型语言，运行代码只需要一个解释器，不需要编译。

## 91. 谈一下什么是解释性语言，什么是编译性语言？

计算机不能直接理解高级语言，只能直接理解机器语言，所以必须要把高级语言翻译成机器语言，计算机才能执行高级语言编写的程序。

解释性语言在运行程序的时候才会进行翻译。

编译型语言写的程序在执行之前，需要一个专门的编译过程，把程序编译成机器语言（可执行文件）。

## 92. Python 中有日志吗?怎么使用?

有日志。

Python 自带 logging 模块，调用 logging.basicConfig()方法，配置需要的日志等级和相应的参数， Python 解释器会按照配置的参数生成相应的日志。

## 93. Python 是如何进行类型转换的？

内建函数封装了各种转换函数，可以使用目标类型关键字强制类型转换，进制之间的转换可以用 int('str'，base='n')将特定进制的字符串转换为十进制，再用相应的进制转换函数将十进制转换为目标进制。

可以使用内置函数直接转换的有：

list---->tuple　tuple(list)

tuple---->list　list(tuple)

## 94.工具安装问题

windows 环境

Python2 无法安装 mysqlclient。Python3 无法安装 MySQL-python、 flup、functools32、Gooey、Pywin32、webencodings。matplotlib 在 python3 环境中安装报错：The following required packages can not be

built:freetype, png。需要手动下载安装源码包安装解决。

scipy 在 Python3 环境中安装报错，numpy.distutils.system_info.NotFoundError，需要自己手工下载对应的安装包，依赖 numpy,pandas 必须严格根据 python 版本、操作系统、64 位与否。运行 matplotlib 后发现基础包 numpy+mkl 安装失败，需要自己下载，国内暂无下载源

centos 环境下

Python2 无法安装 mysql-python 和 mysqlclient 包，报错：EnvironmentError: mysql_config not found，解决方案是安装 mysql-devel 包解决。使用 matplotlib 报错：no module named _tkinter，安装 Tkinter、tk-devel、tc-devel 解决。 pywin32 也无法在 centos 环境下安装。

## 95.关于 Python 程序的运行方面，有什么手段能提升性能？

使用多进程，充分利用机器的多核性能

对于性能影响较大的部分代码，可以使用 C 或 C++编写

对于 IO 阻塞造成的性能影响，可以使用 IO 多路复用来解决

尽量使用 Python 的内建函数

尽量使用局部变量

## 96.Python 中的作用域？

Python 中，一个变量的作用域总是由在代码中被赋值的地方所决定。当 Python 遇到一个变量的话它会按照这的顺序进行搜索：

本地作用域(Local)--->当前作用域被嵌入的本地作用域(Enclosing locals)--->全局/模块作用域(Global)--->内置作用域(Built-in)。

## 97.什么是 Python？

- Python 是一种编程语言，它有对象、模块、线程、异常处理和自动内存管理，可以加入其他语言的对比。
- Python 是一种解释型语言，Python 在代码运行之前不需要解释。
- Python 是动态类型语言，在声明变量时，不需要说明变量的类型。
- Python 适合面向对象的编程，因为它支持通过组合与继承的方式定义类。
- 在 Python 语言中，函数是第一类对象。
- Python 代码编写快，但是运行速度比编译型语言通常要慢。
- Python 用途广泛，常被用走"胶水语言"，可帮助其他语言和组件改善运行状况。
- 使用 Python，程序员可以专注于算法和数据结构的设计，而不用处理底层的细节。

## 98.什么是 Python 的命名空间？

在 Python 中，所有的名字都存在于一个空间中，它们在该空间中存在和被操作——这就是命名空间。它就好像一个盒子，每一个变量名字都对应装着一个对象。当查询变量的时候，会从该盒子里面寻找相应的对象。

## 99.你所遵循的代码规范是什么？请举例说明其要求？

PEP8 规范。

1. 变量

常量：大写加下划线 USER_CONSTANT。

私有变量 ：小写和一个前导下划线 _private_value。

Python 中不存在私有变量一说，若是遇到需要保护的变量，使用小写和一个前导下划线。但这只是程序员之间的一个约定，用于警告说明这是一个私有变量，外部类不要去访问它。但实际上，外部类还是可以访问到这个变量。

内置变量 ：小写，两个前导下划线和两个后置下划线 __class__ 两个前导下划线会导致变量在解释期间被更名。这是为了避免内置变量和其他变量产生冲突。用户定义的变量要严格避免这种风格。以免导致混乱。

2. 函数和方法

总体而言应该使用，小写和下划线。但有些比较老的库使用的是混合大小写，即首单词小写，之后每个单词第一个字母大写，其余小写。但现在，小写和下划线已成为规范。

私有方法 ：小写和一个前导下划线

这里和私有变量一样，并不是真正的私有访问权限。同时也应该注意一般函数不要使用两个前导下划线(当遇到两个前导下划线时，Python 的名称改编特性将发挥作用)。

特殊方法 ：小写和两个前导下划线，两个后置下划线这种风格只应用于特殊函数，比如操作符重载等。

函数参数 ：小写和下划线，缺省值等号两边无空格

3. 类

类总是使用驼峰格式命名，即所有单词首字母大写其余字母小写。类名应该简明，精确，并足以从中理解类所完成的工作。常见的一个方法是使用表示其类型或者特性的后缀，例如:

SQLEngine，MimeTypes 对于基类而言，可以使用一个 Base 或者 Abstract 前缀 BaseCookie，AbstractGroup

4. 模块和包

除特殊模块 __init__ 之外，模块名称都使用不带下划线的小写字母。

若是它们实现一个协议，那么通常使用 lib 为后缀，例如:

```
import smtplib
import os
import sys
```

5. 关于参数

不要用断言来实现静态类型检测。断言可以用于检查参数，但不应仅仅是进行静态类型检测。

Python 是动态类型语言，静态类型检测违背了其设计思想。断言应该用于避免函数不被毫无意义的调用。

不要滥用 *args 和 **kwargs。*args 和 **kwargs 参数可能会破坏函数的健壮性。它们使签名变得模糊，而且代码常常开始在不应该的地方构建小的参数解析器。

6. 其他

使用 has 或 is 前缀命名布尔元素 is_connect = True

has_member = False

用复数形式命名序列

members = ['user_1', 'user_2']

用显式名称命名字典

person_address = {'user_1':'10 road WD', 'user_2' : '20 street huafu'}

避免通用名称

诸如 list, dict, sequence 或者 element 这样的名称应该避免。

避免现有名称诸如 os, sys 这种系统已经存在的名称应该避免。

7. 一些数字

一行列数 : PEP 8 规定为 79 列。根据自己的情况，比如不要超过满屏时编辑器的显示列数。

一个函数 : 不要超过 30 行代码, 即可显示在一个屏幕类，可以不使用垂直游标即可看到整个函数。

一个类 : 不要超过 200 行代码，不要有超过 10 个方法。一个模块 不要超过 500 行。

8. 验证脚本可以安装一个 pep8 脚本用于验证你的代码风格是否符合 PEP8。

# 十七、 Python2 与 Python3 的区别

## 100. 核心类差异

1.Python3 对 Unicode 字符的原生支持。

Python2 中使用 ASCII 码作为默认编码方式导致 string 有两种类型 str 和 unicode，Python3 只支持 unicode 的 string。Python2 和 Python3 字节和字符对应关系为：

| python2 | python3 | 表现 | 转换 | 作用 |
|---|---|---|---|---|
| str | bytes | 字节 | encode | 存储 |
| unicode | str | 字符 | decode | 显示 |

2.Python3 采用的是绝对路径的方式进行 import。

Python2 中相对路径的 import 会导致标准库导入变得困难（想象一下，同一目录下有 file.py，如何同时导入这个文件和标准库 file）。Python3 中这一点将被修改，如果还需要导入同一目录的文件必须使用绝对路

径，否则只能使用相关导入的方式来进行导入。

3. Python2 中存在老式类和新式类的区别，Python3 统一采用新式类。新式类声明要求继承 object，必须用新式类应用多重继承。

4. 缩进严格程度

Python3 使用更加严格的缩进。Python2 的缩进机制中，1 个 tab 和 8 个 space 是等价的，所以在缩进中可以同时允许 tab 和 space 在代码中共存。这种等价机制会导致部分 IDE 使用存在问题。

Python3 中 1 个 tab 只能找另外一个 tab 替代，因此 tab 和 space 共存会导致报错：TabError: inconsistent use of tabs and spaces in indentation.

## 101. 废弃类差异

1. print 语句被 Python3 废弃，统一使用 print 函数

2. exec 语句被 python3 废弃，统一使用 exec 函数

3. execfile 语句被 Python3 废弃，推荐使用 exec(open("./filename").read())

4. 不相等操作符"<>"被 Python3 废弃，统一使用"!="

5. long 整数类型被 Python3 废弃，统一使用 int

6. xrange 函数被 Python3 废弃，统一使用 range，Python3 中 range 的机制也进行修改并提高了大数据集生成效率

7. Python3 中这些方法再不再返回 list 对象：dictionary 关联的 keys()、values()、items()，zip()，map()，filter()，但是可以通过 list 强行转换：

8. mydict={"a":1,"b":2,"c":3}

9. mydict.keys() #<built-in method keys of dict object at 0x000000000040B4C8>

10. list(mydict.keys()) #['a', 'c', 'b']

11. 迭代器 iterator 的 next()函数被 Python3 废弃，统一使用 next(iterator)

12. raw_input 函数被 Python3 废弃，统一使用 input 函数

13. 字典变量的 has_key 函数被 Python 废弃，统一使用 in 关键词

14. file 函数被 Python3 废弃，统一使用 open 来处理文件，可以通过 io.IOBase 检查文件类型

15. apply 函数被 Python3 废弃

16. 异常 StandardError 被 Python3 废弃，统一使用 Exception

## 102. 修改类差异

1.浮点数除法操作符“/”和“//”的区别

“/”：

Python2：若为两个整形数进行运算，结果为整形，但若两个数中有一个为浮点数，则结果为浮点数；

Python3:为真除法，运算结果不再根据参加运算的数的类型。

“//”：

Python2：返回小于除法运算结果的最大整数；从类型上讲，与"/"运算符返回类型逻辑一致。

Python3：和 Python2 运算结果一样。

2.异常抛出和捕捉机制区别

Python2

```
raise IOError, "file error" #抛出异常
except NameError, err:  #捕捉异常
```

Python3

```
raise IOError("file error") #抛出异常
```

except NameError as err: #捕捉异常

3.for 循环中变量值区别

Python2，for 循环会修改外部相同名称变量的值

```
i = 1
print （'comprehension: ', [i for i in range(5)]）
print （'after: i =', i） #i=4
```

Python3，for 循环不会修改外部相同名称变量的值

```
i = 1
print （'comprehension: ', [i for i in range(5)]）
print （'after: i =', i ） #i=1
```

4.round 函数返回值区别 Python2，round 函数返回 float 类型值

```
isinstance(round(15.5),int) #True
```

Python3，round 函数返回 int 类型值

```
isinstance(round(15.5),float) #True
```

5.比较操作符区别

Python2 中任意两个对象都可以比较

```
11 < 'test' #True
```

Python3 中只有同一数据类型的对象可以比较

```
11 < 'test' # TypeError: unorderable types: int() < str()
```

## 103. 第三方工具包差异

我们在 pip 官方下载源 **pypi** 搜索 Python2.7 和 Python3.5 的第三方工具包数可以发现，Python2.7 版本对应的第三方工具类目数量是 28523,Python3.5 版本的数量是 12457，这两个版本在第三方工具包支持数量差距相当大。

我们从数据分析的应用角度列举了常见实用的第三方工具包（如下表），并分析这些工具包在 Python2.7 和 Python3.5 的支持情况：

| | 工具名 | 用途 |
|---|---|---|
| 数据收集 | scrapy | 网页采集，爬虫 |
| 数据收集 | scrapy-redis | 分布式爬虫 |
| 数据收集 | selenium | web 测试，仿真浏览器 |
| 数据处理 | beautifulsoup | 网页解释库，提供 lxml 的支持 |
| 数据处理 | lxml | xml 解释库 |
| 数据处理 | xlrd | excel 文件读取 |
| 数据处理 | xlwt | excel 文件写入 |
| 数据处理 | xlutils | excel 文件简单格式修改 |
| 数据处理 | pywin32 | excel 文件的读取写入及复杂格式定制 |
| 数据处理 | Python-docx | Word 文件的读取写入 |
| 数据分析 | numpy | 基于矩阵的数学计算库 |
| 数据分析 | pandas | 基于表格的统计分析库 |

| **分类** | **工具名** | **用途** |
|---|---|---|
| 数据分析 | scipy | 科学计算库，支持高阶抽象和复杂模型 |
| 数据分析 | statsmodels | 统计建模和计量经济学工具包 |

| | | |
|---|---|---|
| 数据分析 | scikit-learn | 机器学习工具库 |
| 数据分析 | gensim | 自然语言处理工具库 |
| 数据分析 | jieba | 中文分词工具库 |
| 数据存储 | MySQL-python | mysql 的读写接口库 |
| 数据存储 | mysqlclient | mysql 的读写接口库 |
| 数据存储 | SQLAlchemy | 数据库的 ORM 封装 |
| 数据存储 | pymssql | sql server 读写接口库 |
| 数据存储 | redis | redis 的读写接口 |
| 数据存储 | PyMongo | mongodb 的读写接口 |
| 数据呈现 | matplotlib | 流行的数据可视化库 |
| 数据呈现 | seaborn | 美观的数据可是湖库，基于 matplotlib |
| 工具辅助 | jupyter | 基于 web 的 python IDE，常用于数据分析 |
| 工具辅助 | chardet | 字符检查工具 |
| 工具辅助 | ConfigParser | 配置文件读写支持 |
| 工具辅助 | requests | HTTP 库，用于网络访问 |

# 第十章　Selenium 相关

## 1. 官方文档地址

https://seleniumhq.github.io/selenium/docs/api/py/api.html

## 2. 常用自动化测试工具机器运行原理，写出一段元素查找的代码？

webdriver 原理：

- 每个 Selenium 命令，这里指的是所谓的基础操作，例如，点击、输入等，都会创建一条 HTTP 请求，发送给 Browser WebDriver
- Browser WebDriver 使用一个 HTTP Server 监听和接收 HTTP 请求
- HTTP Server 根据协议规则定义这些 Selenium 命令对应的浏览器具体操作
- 浏览器执行这些操作
- 浏览器将执行状态返回给 HTTP Server
- HTTP Server 再将这些状态信息返回给自动化脚本

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

driver = webdriver.Chrome()
driver.get("http://www.python.org")
assert "Python" in driver.title
elem = driver.find_element_by_name("q")
elem.send_keys("pycon")
elem.send_keys(Keys.RETURN)
assert "No result found." not in driver.page_source
driver.close()
```

## 3. 如何开展自动化测试框架的构建？

我们公司的自动化测试框架主要是有页面库，数据驱动，测试脚本，测试报告，持续集成这几个部分组成的。

页面对象库对自动化包括工具（selenium，appium）API 的二次封装，还有使用二次封装后的自动化工具类实现的页面元素封装（Page Object）然后会给封装好的页面设置一个统一入口类。这些之中会有一个页面

元素文件专门存放元素的定位方法。

数据驱动部分主要是测试脚本中使用的数据文件（excel,yaml,txt）以及读取方法类，如果数据涉及到数据库，也会把对应的数据读取方法封装到这个部分。

测试脚本主要是通过 pytest 测试框架进行编写的，选择其的原因主要有其支持 assert 语句断言，适合复杂的功能测试，执行过程中可以自定义用例执行顺序和跳过以及预期，支持重复执行，还可兼容 unittest 编写的测试用例，最重要的是支持参数化和方便持续集成工具集成。

测试报告主要是通过 pytest 自动生成的 Allure 报告，其可读性可生动的数据表图比 pytest 报告更能反应测试结果，也可以集成与 Jenkins 中。

持续集成方面主要是通过 Jenkins 进行实现的，目的在于测试脚本的无人值守执行以及自动生成测试报告，方便测试人员能够省出时间进行更多的功能测试和探索性测试。（通过设置几个 git,gitlab，mailer,allure，等功能插件，配置 Allure 报告，默认邮件发送设置。用例脚本主要存放在 gitlab 用例库中，设置好轮询策略之后，配置报告发送的目标邮箱，就可以实现持续集成实践中的测试环节）

## 4. 如何设计自动化测试用例:

- 编写测试脚本之前要编写测试用例，而且测试用例不能直接使用手工测试的用例。
- 自动化的测试用例是一个完整的场景。用户登录系统到用户退出。
- 用例之验证一个功能点。不用试图登陆后验证所有的功能在退出
- 测试用例尽量只做正向的逻辑验证。
- 用例之间不要产生关联，相互独立，也要高内聚，低耦合
- 测试用例关注的是功能逻辑的实现，字段无关
- 测试用例的上下文必须有一定的顺序性，前置条件清晰
- 检查点的设置要侧重，全面，灵活
- 测试用例对数据的操作要进行还原
- 测试用例必须是可回归的
- 用例选择遵循成本始终，构建场景，目的冒烟回归，繁琐功能，主体流程
- 用例转型遵循前置配置，抛异常，步骤验证，高内聚，关门归原

## 5. webdriver 如何开启和退出一个浏览器？

开启：dr = webdriver.浏览器类型()

关闭：dr.quit()

# 第十一章　Jmeter 相关

## 1. Jmeter 的七大原件是什么？有什么作用？

2.3.2 JMeter 主要测试元件

Test Plan (测试计划)：用于描述包含与此性能测试相关的所有相关功能的性能测试。换句话说，性能测试的所有内容都是基于一个计划。

Threads（Users）线程用户：一个线程组，可以看作是一个虚拟用户组，线程组中的每个线程都可以理解为一个虚拟用户。 在测试执行期间，线程组中包含的线程数量没有变化。

测试片段（Test Fragment）：测试片段是控制器上的特殊线程组，它与测试树上的线程组处于层次结构中。 它与线程组不同，因为它不会执行，除非它是一个模块控制器或被控制器引用。

取样器（Sampler）：取样器是性能测试向服务器发送请求以记录响应信息，记录响应时间的最小单位，JMeter 本机支持各种不同的采样器。

逻辑控制器（Logic Controller）：逻辑控制器由两个组件组成，一个是用于控制采样器节点的逻辑顺序的控制器，在计划中发送测试请求，另一个用于组织可以控制采样器到节点。

配置元件（config element）：用于提供对静态数据配置的支持。

定时器（Timer）：设置操作之间的等待时间，等待时间通常用于控制客户端 QPS 的性能。

前置处理器（Per Processors）：用于在发出实际请求之前发送的请求的特殊处理。

后置处理器（Post Processors）：用于向采样器服务器发送请求以响应响应。

断言（Assertions）：用于检查测试中得到的相应数据等是否符合预期。

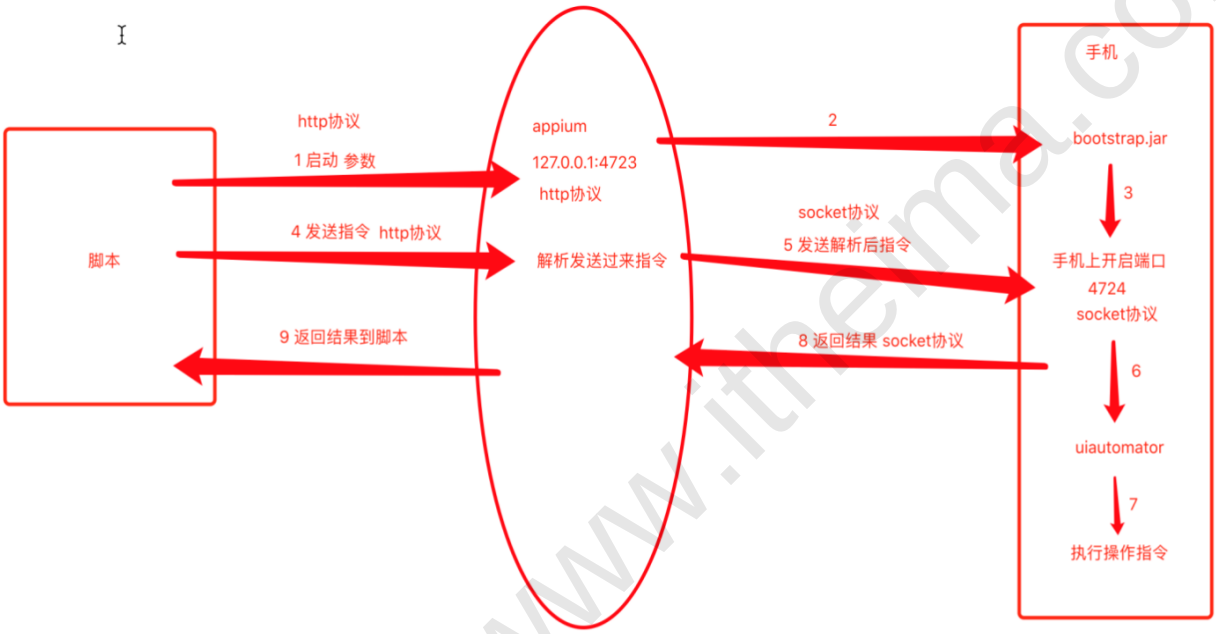
监听器（Listener）：用于对测试结果数据进行处理和可视化展示的一系列元件。

## 2. 聚合报告的每个字段代表的是什么意思？

## 3. 写一个验证电子邮件格式的正则表达式？

## 4. 一台客户端有 500 个客户与 500 个客户端有 300 个用户对服务器施压，有什么区别？

# 第十二章　Appium 相关




1. **Appium 的运行原理是什么？**

# 第十三章　性能测试

## 1. 常见性能测试的方法有哪些？举例解释一下？

1．负载测试

在这里，负载测试指的是最常见的验证一般性能需求而进行的性能测试，在上面我们提到了用户最常见的性能需求就是“既要马儿跑，又要马儿少吃草”。因此负载测试主要是考察软件系统在既定负载下的性能表现。我们对负载测试可以有如下理解：

（1）负载测试是站在用户的角度去观察在一定条件下软件系统的性能表现。

（2）负载测试的预期结果是用户的性能需求得到满足。此指标一般体现为响应时间、交易容量、并发容量、资源使用率等。

2．压力测试

压力测试是为了考察系统在**条件下的表现，**条件可以是超负荷的交易量和并发用户数。注意，这个**条件并不一定是用户的性能需求，可能要远远高于用户的性能需求。可以这样理解，压力测试和负载测试不同的是，压力测试的预期结果就是系统出现问题，而我们要考察的是系统处理问题的方式。比如说，我们期待一个系统在面临压力的情况下能够保持稳定，处理速度可以变慢，但不能系统崩溃。因此，压力测试是能让我们识别系统的弱点和在极限负载下程序将如何运行。

例子：负载测试关心的是用户规则和需求，压力测试关心的是软件系统本身。

3．并发测试

验证系统的并发处理能力。一般是和服务器端建立大量的并发连接，通过客户端的响应时间和服务器端的性能监测情况来判断系统是否达到了既定的并发能力指标。负载测试往往就会使用并发来创造负载，之所以把并发测试单独提出来，是因为并发测试往往涉及服务器的并发容量，以及多进程/多线程协调同步可能带来的问题。这是要特别注意，必须测试的。

4．基准测试

当软件系统中增加一个新的模块的时候，需要做基准测试，以判断新模块对整个软件系统的性能影响。按照基准测试的方法，需要打开/关闭新模块至少各做一次测试。关闭模块之前的系统各个性能指标记下来作为基准（Benchmark），然后与打开模块状态下的系统性能指标作比较，以判断模块对系统性能的影响。

5．稳定性测试

“路遥知马力”，在这里我们要说的是和性能测试有关的稳定性测试，即测试系统在一定负载下运行长时间后是否会发生问题。软件系统的有些问题是不能一下子就暴露出来的，或者说是需要时间积累才能达到能够度量的程度。为什么会需要这样的测试呢？因为有些软件的问题只有在运行一天或一个星期甚至更长的时间才会暴露。这种问题一般是程序占用资源却不能及时释放而引起的。比如，内存泄漏问题就是经过一段时间积累才会慢慢变得显著，在运行初期却很难检测出来；还有客户端和服务器在负载运行一段时间后，建立了大量的连接通路，却不能有效地复用或及时释放。

6．可恢复测试

测试系统能否快速地从错误状态中恢复到正常状态。比如，在一个配有负载均衡的系统中，主机承受了压力无法正常工作后，备份机是否能够快速地接管负载。可恢复测试通常结合压力测试一起来做。

## 2. 你认为性能测试的目的是什么？做好性能测试的工作的关键是什么？

性能测试工作的目的是检查系统是否满足在需求说明书中规定的性能，性能测试常常需要和强度测试结合起来，并常常要求同时进行软件和硬件的检测。

性能测试主要的关注对象是响应时间，吞吐量，占用内存大小（辅助存储区），处理精度等。

## 3. 服务端性能分析都从哪些角度来进行？

从维度上划分，性能指标主要分为两大类，分别是业务性能指标和系统资源性能指标。

业务性能指标可以直观地反映被测系统的实际性能状况，常用的指标项有：

1.并发用户数

2.事务吞吐率（TPS/RPS）

3.事务平均响应时间

4.事务成功率

系统资源性能指标，主要是反映整个系统环境的硬件资源使用情况，常用的指标包括：

1.服务器：CPU 利用率、处理器队列长度、内存利用率、内存交换页面数、磁盘 IO 状态、网卡带宽使用情况等；

2.数据库：数据库连接数、数据库读写响应时长、数据库读写吞吐量等；

3.网络：网络吞吐量、网络带宽、网络缓冲池大小；

4.缓存（Redis）：静态资源缓存命中率、动态数据缓存命中率、缓存吞吐量等；

5.测试设备（压力发生器）：CPU 利用率、处理器队列长度、内存利用率、内存交换页面数、磁盘 IO 状态、网卡带宽使用情况等。

如何理解压力测试，负载测试以及性能测试？

## 4. 如何理解压力测试，负载测试以及性能测试？

性能测试（Performance Test）：通常收集所有和测试有关的所有性能，被不同人在不同场合下进行使用。

压力测试 stress test：是在一定的『负荷条件』下，长时间连续运行系统给系统性能造成的影响。

负载测试 Load test：在一定的『工作负荷』下，给系统造成的负荷及系统响应的时间。

## 5. 编写一个 http 接口性能测试方案，测试过程的关注点有哪些，流程等？

一、准备工作

1、系统基础功能验证

性能测试在什么阶段适合实施？切入点很重要！一般而言，只有在系统基础功能测试验证完成、系统趋于稳定的情况下，才会进行性能测试，否则性能测试是无意义的。

c2、测试团队组建

根据该项目的具体情况，组建一个几人的性能测试 team，其中 DBA 是必不可少的，然后需要一至几名系统开发人员（对应前端、后台等），还有性能测试设计和分析人员、脚本开发和执行人员；在正式开始工作之前，应该对脚本开发和执行人员进行一些培训，或者应该由具有相关经验的人员担任。

3、工具的选择

综合系统设计、工具成本、测试团队的技能来考虑，选择合适的测试工具，最起码应该满足一下几点：支持对 web（这里以 web 系统为例）系统的性能测试，支持 http 和 https 协议；工具运行在 Windows 平台上；支持对 webserver、前端、数据库的性能计数器进行监控；

4、预先的业务场景分析

为了对系统性能建立直观上的认识和分析，应对系统较重要和常用的业务场景模块进行分析，针对性的进行分析，以对接下来的测试计划设计进行准备。

二、测试计划

测试计划阶段最重要的是分析用户场景，确定系统性能目标。

1、性能测试领域分析

根据对项目背景，业务的了解，确定本次性能测试要解决的问题点；是测试系统能否满足实际运行时的需要，还是目前的系统在哪些方面制约系统性能的表现，或者，哪些系统因素导致系统无法跟上业务发展？确定测试领域，然后具体问题具体分析。

2、用户场景剖析和业务建模

根据对系统业务、用户活跃时间、访问频率、场景交互等各方面的分析，整理一个业务场景表，当然其中最好对用户操作场景、步骤进行详细的描述，为测试脚本开发提供依据。

3、确定性能目标

前面已经确定了本次性能测试的应用领域，接下来就是针对具体的领域关注点，确定性能目标（指标）；其中需要和其他业务部门进行沟通协商，以及结合当前系统的响应时间等数据，确定最终我们需要达到的响应时间和系统资源使用率等目标；比如：登录请求到登录成功的页面响应时间不能超过 2 秒；报表审核提交的页面响应时间不能超过 5 秒；文件的上传、下载页面响应时间不超过 8 秒；服务器的 CPU 平均使用率小于 70%，内存使用率小于 75%；各个业务系统的响应时间和服务器资源使用情况在不同测试环境下，各指标随负载变化的情况等；

4、制定测试计划的实施时间

预设本次性能测试各子模块的起止时间，产出，参与人员等等。

三、测试脚本设计与开发

性能测试中，测试脚本设计与开发占据了很大的时间比重。

1、测试环境设计

本次性能测试的目标是需要验证系统在实际运行环境中的性能外，还需要考虑到不同的硬件配置是否会是制约系统性能的重要因素！因此在测试环境中，需要部署多个不同的测试环境，在不同的硬件配置上检查应用系统的性能，并对不同配置下系统的测试结果进行分析，得出最优结果（最适合当前系统的配置）。

这里所说的配置大概是如下几类：数据库服务器;应用服务器;负载模拟器;软件运行环境，平台。

测试环境测试数据，可以根据系统的运行预期来确定，比如需要测试的业务场景，数据多久执行一次备份转移，该业务场景涉及哪些表，每次操作数据怎样写入，写入几条，需要多少的测试数据来使得测试环境的数据保持一致性等等。可以在首次测试数据生成时，将其导出到本地保存，在每次测试开始前导入数据，保持一致性。

2、测试场景设计

通过和业务部门沟通以及以往用户操作习惯，确定用户操作习惯模式，以及不同的场景用户数量，操作次数，确定测试指标，以及性能监控等。

3、测试用例设计

确认测试场景后，在系统已有的操作描述上，进一步完善为可映射为脚本的测试用例描述，用例大概内容如下：

用例编号：查询表单_xxx_x1（命名以业务操作场景为主，简洁易懂即可）

用例条件：用户已登录、具有对应权限等

操作步骤：系统业务场景描述

4、脚本和辅助工具的开发及使用

按照用例描述，可利用工具进行录制，然后在录制的脚本中进行修改；比如参数化、关联、检查点等等，最后的结果使得测试脚本可用，能达到测试要求即可；建议尽量自己写脚本来实现业务操作场景，这样对个人技能提升较大；一句话：能写就绝不录制!!!

四、测试执行与管理

在这个阶段，只需要按照之前已经设计好的业务场景、环境和测试用例脚本，部署环境，执行测试并记录结果即可。

1、建立测试环境

按照之前已经设计好的测试环境，部署对应的环境，由运维或开发人员进行部署，检查，并仔细调整，同时保持测试环境的干净和稳定，不受外来因素影响。

2、执行测试脚本

这一点比较简单，在已部署好的测试环境中，按照业务场景和编号，按顺序执行我们已经设计好的测试脚本。

3、测试结果记录

根据测试采用的工具不同，结果的记录也有不同的形式；现在大多的性能测试工具都提供比较完整的界面图形化的测试结果，当然，对于服务器的资源使用等情况，可以利用一些计数器或第三方监控工具来对其进行记录，执行完测试后，对结果进行整理分析。

五、测试分析

1、测试环境的系统性能分析

根据我们之前记录得到的测试结果（图表、曲线等），经过计算，与预定的性能指标进行对比，确定是否达到了我们需要的结果；如未达到，查看具体的瓶颈点，然后根据瓶颈点的具体数据，进行具体情况具体分析（影响性能的因素很多，这一点，可以根据经验和数据表现来判断分析）。

2、硬件设备对系统性能表现的影响分析

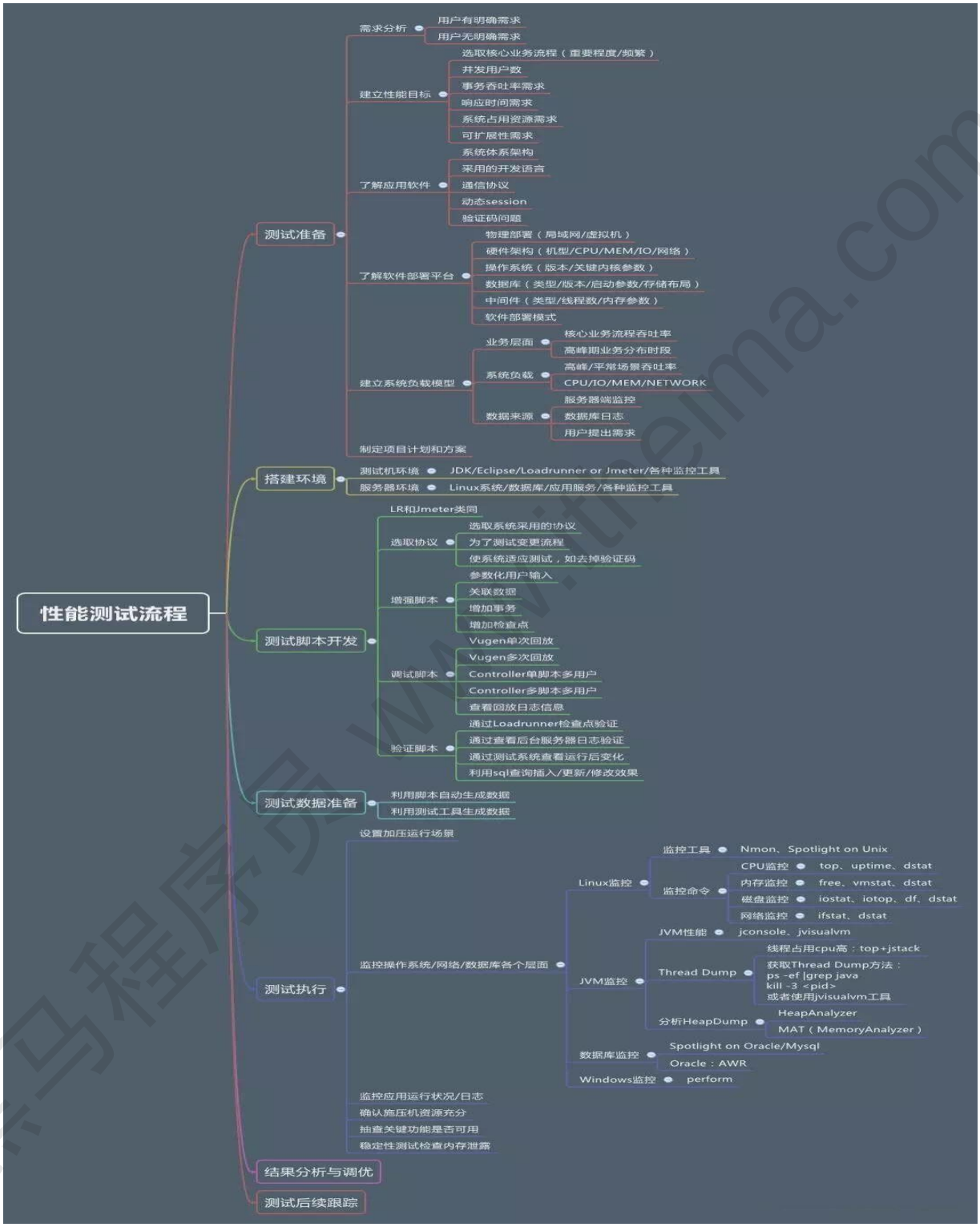
由于之前设计了几个不同的测试环境，故可以根据不同测试环境的硬件资源使用状况图进行分析，确定瓶颈是在数据库服务器、应用服务器抑或其他方面，然后针对性的进行优化等操作。

3、其他影响因素分析

影响系统性能的因素很多，可以从用户能感受到的场景分析，哪里比较慢，哪里速度尚可，这里可以根据 2\5\8 原则对其进行分析；至于其他诸如网络带宽、操作动作、存储池、线程实现、服务器处理机制等一系列的影响因素，具体问题具体分析，这里就不一一表述了。

4、测试中发现的问题

在性能测试执行过程中，可能会发现某些功能上的不足或存在的缺陷，以及需要优化的地方，这也是执行多次测试的优点。




**6. 如何判断是否有内存泄漏及关注的指标？**

# 第十四章 LordRunner 相关

## 1. LoadRunner 的工作原理是什么？

LoadRunner 工作原理：

LoadRunner 通过模拟上千万用户实施并发负载，实时性能监控的系统行为和性能方式来确认和查找问题。

1、VuGen 发生器：捕捉用户的业务流，并最终将其录制成一个脚本：

（1）选择相应的一种协议；

（2）在客户端模拟用户使用过程中的业务流程，并录制成一个脚本；

（3）编辑脚本和设置 Run-Time Settings 项；

（4）编译脚本生成一个没有错误的可运行的脚本。

2、控制器（Controller）：

（1）设计场景，包括手动场景设计和目标场景设计两种方式；

（2）场景监控，可以实时监控脚本的运行的情况。可以通过添加计数器来监控 Windows 资源、应用服务器和数据库使用情况。

场景设计的目的是设计出一个最接近用户实际使用的场景，场景设计越接近用户使用的实际情况，测试出来的数据就越接近真实值。

3、负载发生器（Load Generators）：模拟用户对服务器提交请求。

通常，在性能测试过程中会将控制器和负载发生器分开；当使用多台负载发生器时，一定要保证负载均衡（指在进行性能测试的过程中，保证每台负载发生器均匀地对服务器进行施压）。

4、分析器（Analysis）：主要用于对测试结果进行分析。

## 2. LoadRunner 脚本如何录制和编写？

打开 loadrunner 的 Virtual User Generator，新建脚本

在弹出框中选择 Web（HTTP/HTML）协议，然后点击创建按钮

弹出 start Recording 窗口，选择对应的录制类型（Internet Applications），选择浏览器（这里我们选择 IE），选择需要测试的 web 地址，选择浏览器安装地址。点击 ok

自动打开 IE 浏览器，进入相对应地址，在页面上方显示一个录制工具条。此时我们发给服务器的所有请

求都会被记录在脚本中。输入用户名和密码，在点击登录前插入事务，输入事务名称，点击 ok

然后点击登录按钮，待登录成功，显示出成功页面后，点击结束事务，再点击 ok。然后点击工具条上的停止按钮。结束录制，回到脚本中。这时候需要等待会，待自动生成脚本。

生成的脚本含有刚才录制的信息，点击菜单栏，回放按钮。回放如果有红色，是报错信息，没有红色，如下图，说明运行成功。

还可点击“View”菜单栏的“Test Results”进行查看。显示 passed 即为成功。脚本便可使用。

## 3. LoadRunner 中的 Think Time 有什么作用？

用户在执行连续操作之间等待的时间称为“思考时间”，它是决定对服务器施压大小的因素之一。设置思考时间，是为了更真实的模拟用户。Vuser 使用 Lr_think_time 函数来模拟用户思考时间。录制 Vuser 脚本时，VuGen 将录制实际思考时间，并插入到 Vuser 脚本中响应的 Lr_think_time 语句。可以编辑录制的 Lr_think_time 语句，并向 Vuser 脚本手动添加更多 Lr_think_time 语句。

可以通过选择【插入】>【步骤】>【思考时间】来插入思考时间步骤。当录制 Java Vuser 脚本时，不会在 Vuser 脚本中生成 Lr_think_time 语句。

可以使用【Run-time Settings】，更改执行 Vuser 脚本时 Lr_think_time 语句的运行方式。

## 4. 在搜索引擎中输入汉字就可以解析到对应的域名，请问如何用 LoadRunner 进行测试？

a)建立测试计划，确定测试标准和测试范围

b)设计典型场景的测试用例，覆盖常用业务流程和不常用的业务流程等

c)根据测试用例，开发自动测试脚本和场景：

i.录制测试脚本

1.新建一个脚本（Web/HTML 协议）

2.点击录制按钮，在弹出的对话框的 URL 中输入”about:blank”。

3.在打开的浏览器中进行正常操作流程后，结束录制。

4.调试脚本并保存。可能要注意到字符集的关联。

ii.设置测试场景

1.针对性能设置测试场景，主要判断在正常情况下，系统的平均事务响应时间是否达标

2.针对压力负载设置测试场景，主要判断在长时间处于满负荷或者超出系统承载能力的条件下，系统是否会崩溃。

iii.执行测试，获取测试结果，分析测试结果

**5. 一台客户端有三百个客户与三百个客户端有三百个客户对服务器施压，有什么区别？**

300 个用户在一个客户端上，会占用客户机更多的资源，而影响测试的结果。

线程之间可能发生干扰，而产生一些异常。

300 个用户在一个客户端上，需要更大的带宽。

IP 地址的问题，可能需要使用 IP Spoof 来绕过服务器对于单一 IP 地址最大连接数的限制。

所有用户在一个客户端上，不必考虑分布式管理的问题；而用户分布在不同的客户端上，需要考虑使用控制器来整体调配不同客户机上的用户。同时，还需要给予相应的权限配置和防火墙设置。

**6. 2018 年春运前第 10 天中午（2018 年 1 月 23 日），12306 服务器挂了大概 30 分钟，工程师抢修以后，马上上线，之后又挂了，请问有哪些原因会造成这个情况？**

**7. 在搜索引擎中输入汉字就可以解析到对应的域名，请问如何用 LoadRunner 进行测试**

**8. 性能压测过程中，当面对大量并发用户调用的时候，服务器端 CPU 的使用率是高好还是低好？为什么？**

**9. 测试某一个接口的压测，需要准备哪些数据，怎么进行参数化，不同的测试压力指标，需要准备多少数据比较好？**

**10. APP 崩溃率居高不下，通过哪些措施来发现这些问题，并后续处理？**

# 第十五章　计算机基础

## 一、 操作系统

### 1. 什么是内存泄漏？什么是内存溢出？二者有什么区别？

内存溢出（OutOfMemory-OOM）：指你的应用的内存已经不能满足正常使用了，堆栈已经达到系统设置的最大值，进而导致崩溃，这是一种结果描述。

内存泄漏（Memory Leak）：指你的应用使用资源之后没有及时释放，导致应用内存中持有了不需要的资源，这是一种状态描述。

### 2. 了解的操作系统有哪些？

Windows，Unix，Linux，Mac

## 二、 计算机网络

### 1. 什么是局域网，广域网？

一、局域网

局域网（Local Area Network），简称 LAN，是指在某一区域内由多台计算机互联成的计算机组。“某一区域”指的是同一办公室、同一建筑物、同一公司和同一学校等，一般是方圆几千米以内。局域网可以实现文件管理、应用软件共享、打印机共享、扫描仪共享、工作组内的日程安排、电子邮件和传真通信服务等功能。局域网是封闭型的，可以由办公室内的两台计算机组成，也可以由一个公司内的上千台计算机组成。

二、广域网

广域网（Wide Area Network），简称 WAN，是一种跨越大的、地域性的计算机网络的集合。通常跨越省、市，甚至一个国家。广域网包括大大小小不同的子网，子网可以是局域网，也可以是小型的广域网。

### 2. 10M 兆宽带是什么意思？理论下载速度是多少？

首先我们要搞懂其中的区别，运营商说的 10M，完整的单位应该是 10Mbps（bps：比特率），而我们讲的下载速度单位是 MB，虽然都念兆，但是不一样的。

它们之间的换算关系是：1MB=8×1Mbps，换个方式看：1Mbps÷8=128KB，也就是说，运营商称的 10M 宽带，实际速度是 10Mbps÷8=1280KB，约 1.25MB。

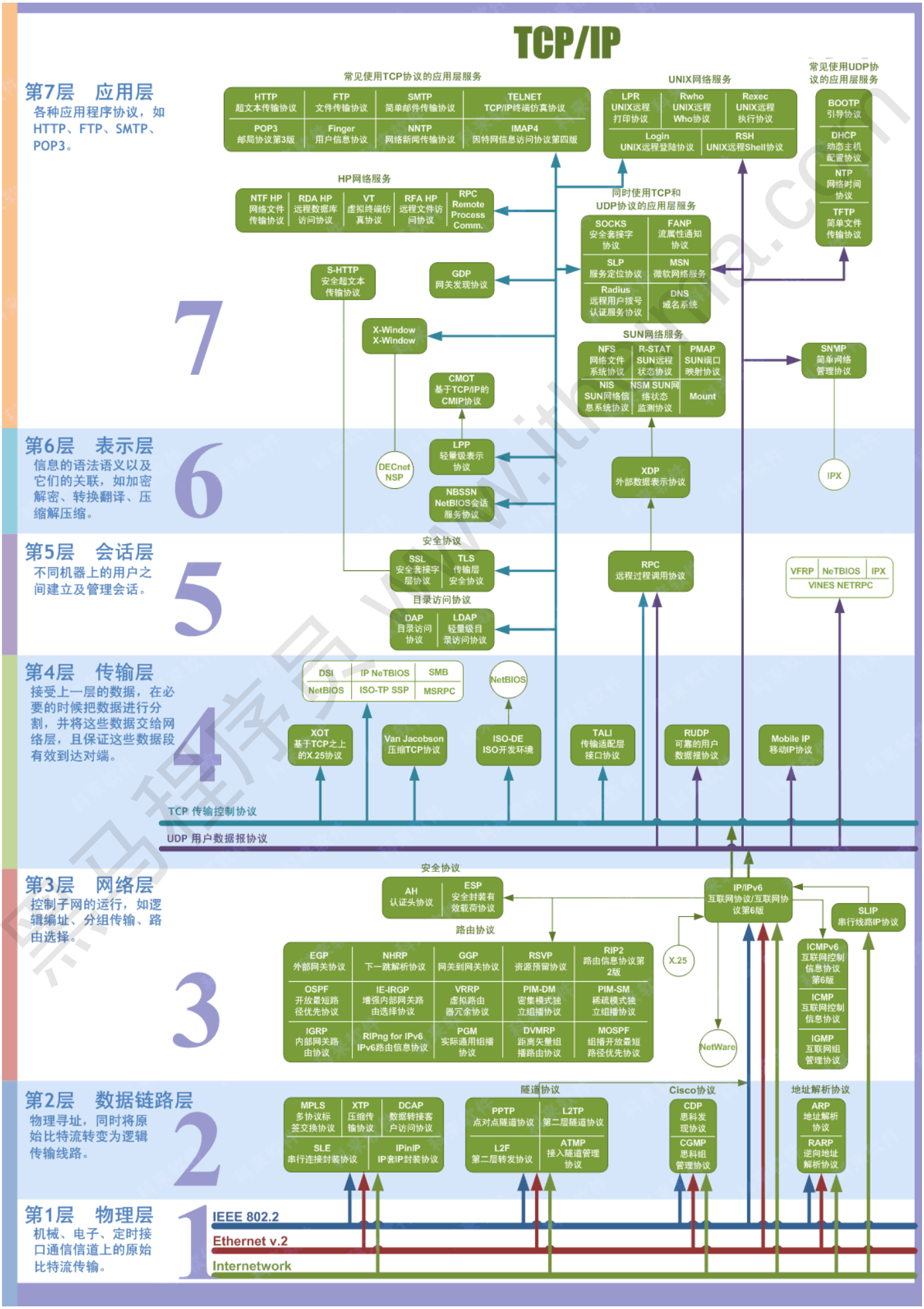
## 3. 什么是 IP 地址？

IP 地址是指互联网协议地址（英语：Internet Protocol Address，又译为网际协议地址），是 IP Address 的缩写。IP 地址是 IP 协议提供的一种统一的地址格式，它为互联网上的每一个网络和每一台主机分配一个逻辑地址，以此来屏蔽物理地址的差异。

**IP地址分类**

| 网络类别 | 最大网络数 | IP地址范围 | 最大主机数 | 私有IP地址范围 |
|---|---|---|---|---|
| A | 126 ( 2^7-2) | 1.0.0.0--126.255.255.255 | 2^24-2 | 10.0.0.0--10.255.255.255 |
| B | 16384(2^14) | 128.0.0.0--191.255.255.255 | 2^16-2 | 172.16.0.0--172.31.255.255 |
| C | 2097152 (2^21) | 192.0.0.0--223.255.255.255 | 2^8-2 | 192.168.0.0--192.168.255.255 |

## 4. OSI 七层网络模型的划分？


TCP/IP
第7层 应用层
各种应用程序协议，如HTTP、FTP、SMTP、POP3。
常见使用TCP协议的应用层服务
HTTP 超文本传输协议
FTP 文件传输协议
SMTP 简单邮件传输协议
TELNET TCP/IP终端仿真协议
POP3 邮局协议第3版
Finger 用户信息协议
NNTP 网络新闻传输协议
IMAP4 因特网信息访问协议第四版
UNIX网络服务
LPR UNIX远程打印协议
Rwho UNIX远程Who协议
Rexec UNIX远程执行协议
Login UNIX远程登陆协议
RSH UNIX远程Shell协议
常见使用UDP协议的应用层服务
BOOTP 引导协议
DHCP 动态主机配置协议
NTP 网络时间协议
TFTP 简单文件传输协议
HP网络服务
NTF HP 网络文件传输协议
RDA HP 远程数据库访问协议
VT 虚拟终端仿真协议
RFA HP 远程文件访问协议
RPC Remote Process Comm.
同时使用TCP和UDP协议的应用层服务
SOCKS 安全套接字协议
FANP 流属性通知协议
SLP 服务定位协议
MSN 微软网络服务
Radius 远程用户拨号认证服务协议
DNS 域名系统
S-HTTP 安全超文本传输协议
GDP 网关发现协议
X-Window X-Window
SUN网络服务
NFS 网络文件系统协议
R-STAT SUN远程状态协议
PMAP SUN端口映射协议
NIS SUN网络信息系统协议
NSM SUN网络状态监测协议
Mount
SNMP 简单网络管理协议
CMOT 基于TCP/IP的CMIP协议
7
第6层 表示层
信息的语法语义以及它们的关联，如加密解密、转换翻译、压缩解压缩。
6
DECnet NSP
LPP 轻量级表示协议
NBSSN NetBIOS会话服务协议
XDP 外部数据表示协议
IPX
第5层 会话层
不同机器上的用户之间建立及管理会话。
5
安全协议
SSL 安全套接字层协议
TLS 传输层安全协议
目录访问协议
DAP 目录访问协议
LDAP 轻量级目录访问协议
RPC 远程过程调用协议
VFRP NeTBIOS IPX VINES NETRPC
第4层 传输层
接受上一层的数据，在必要的时候把数据进行分割，并将这些数据交给网络层，且保证这些数据段有效到达对端。
4
DSI NetBIOS
IP NeTBIOS ISO-TP SSP
SMB MSRPC
NetBIOS
XOT 基于TCP之上的X.25协议
Van Jacobson 压缩TCP协议
ISO-DE ISO开发环境
TALI 传输适配层接口协议
RUDP 可靠的用户数据报协议
Mobile IP 移动IP协议
TCP 传输控制协议
UDP 用户数据报协议
第3层 网络层
控制子网的运行，如逻辑编址、分组传输、路由选择。
3
安全协议
AH 认证头协议
ESP 安全封装有效载荷协议
IP/IPv6 互联网协议/互联网协议第6版
SLIP 串行线路IP协议
路由协议
EGP 外部网关协议
NHRP 下一跳解析协议
GGP 网关到网关协议
RSVP 资源预留协议
RIP2 路由信息协议第2版
X.25
OSPF 开放最短路径优先协议
IE-IRGP 增强内部网关路由选择协议
VRRP 虚拟路由器冗余协议
PIM-DM 密集模式独立组播协议
PIM-SM 稀疏模式独立组播协议
ICMPv6 互联网控制信息协议第6版
ICMP 互联网控制信息协议
IGMP 互联网组管理协议
IGRP 内部网关路由协议
RIPng for IPv6 IPv6路由信息协议
PGM 实际通用组播协议
DVMRP 距离矢量组播路由协议
MOSPF 组播开放最短路径优先协议
NetWare
第2层 数据链路层
物理寻址，同时将原始比特流转变为逻辑传输线路。
2
隧道协议
Cisco协议
地址解析协议
MPLS 多协议标签交换协议
XTP 压缩传输协议
DCAP 数据转接客户访问协议
SLE 串行连接封装协议
IPinIP IP套IP封装协议
PPTP 点对点隧道协议
L2TP 第二层隧道协议
L2F 第二层转发协议
ATMP 接入隧道管理协议
CDP 思科发现协议
CGMP 思科组管理协议
ARP 地址解析协议
RARP 逆向地址解析协议
第1层 物理层
机械、电子、定时接口通信信道上的原始比特流传输。
1
IEEE 802.2
Ethernet v.2
Internetwork

## 5. TCP 和 UDP 有什么不同？

TCP：

优点：可靠 稳定

- TCP 的可靠体现在 TCP 在传输数据之前，会有三次握手来建立连接，而且在数据传递时，有确认. 窗口. 重传. 拥塞控制机制，在数据传完之后，还会断开来连接用来节约系统资源。

缺点：慢，效率低，占用系统资源高，易被攻击

- 在传递数据之前要先建立连接，这会消耗时间，而且在数据传递时，确认机制. 重传机制. 拥塞机制等都会消耗大量时间，而且要在每台设备上维护所有的传输连接。然而，每个连接都会占用系统的 CPU，内存等硬件资源。因为 TCP 有确认机制. 三次握手机制，这些也导致 TCP 容易被利用，实现 DOS. DDOS. CC 等攻击。

UDP：

优点：快，比 TCP 稍安全

- UDP 没有 TCP 拥有的各种机制，是一种无状态的传输协议，所以传输数据非常快，没有 TCP 的这些机制，被攻击利用的机会就少一些，但是也无法避免被攻击。

缺点：不可靠，不稳定

- 因为没有 TCP 的这些机制，UDP 在传输数据时，如果网络质量不好，就会很容易丢包，造成数据的缺失。

适用场景：

- TCP：当对网络质量有要求时，比如 HTTP，HTTPS，FTP 等传输文件的协议；POP，SMTP 等邮件传输的协议
- UDP：对网络通讯质量要求不高时，要求网络通讯速度要快的场景

## 6. HTTP 属于哪一层的协议？

HTTP 协议属于应用层协议

## 7. HTTP 和 HTTPS 的区别？

安全性上的区别:HTTPS：HTTP 协议的安全加强版，通过在 HTTP 上建立加密层，对传输数据进行加密。主要作用可以分为两种：一种是建立一个信息安全通道，来保证数据传输的安全；另一种就是确认网站的真实性。

表现形式：HTTPS 站点会在地址栏上显示一把绿色小锁，表明这是加密过的安全网站，如果采用了全球认证的顶级 EV SSL 证书的话，其地址栏会以绿色高亮显示，方便用户辨认。

SEO:在 2015 年之前百度是无法收录 HTTPS 页面的,不过自从 2015 年 5 月份百度搜索全站 HTTPS 加密后，就已经可以收录 HTTPS 了。谷歌则是从 2014 年起便开始收录 HTTPS 页面，并且 HTTPS 页面权重比 HTTP 页面更高。从 SEO 的角度来说，HTTPS 和 HTTP 区别不大，甚至 HTTPS 效果更好。

技术层面：如果要说 HTTPS 和 HTTP 的区别，最关键的还是在技术层面。比如 HTTP 标准端口是 80，而 HTTPS 标准端口是 443；HTTP 无需证书，HTTPS 需要 CA 机构颁发的 SSL 证书；HTTP 工作于应用层，HTTPS 工作于传输层。

## 8. cookies 和 session 的区别？

cookies:是针对每一个网站的信息，每一个网站只对应一个，其它网站不能访问，这个文件是保存在客户端的，每次你打相应网站，浏览器会查找这个网站的 cookies，如果有就会将这个文件起发送出去。cookies 文件的内容大致包函这些信息如用户名，密码，设置等。

session： 是针对每一个用户的，只有客户机访问，程序就会为这个客户新增一个 session。session 里主要保存的是用户的登录信息，操作信息等。这个 session 在用户访问结束后会被自动消失（如果超时也会）。

## 9. HTTP 的 get 请求和 post 请求的区别？

（1） 在客户端，Get 方式在通过 URL 提交数据，数据在 URL 中可以看到；POST 方式，数据放置在 HTML HEADER 内提交。

（2） GET 方式提交的数据最多只能有 1024 字节，而 POST 则没有此限制。

（3） 安全性问题。正如在（1）中提到，使用 Get 的时候，参数会显示在地址栏上，而 Post 不会。所以，如果这些数据是中文数据而且是非敏感数据，那么使用 get；如果用户输入的数据不是中文字符而且包含敏感数据，那么还是使用 post 为好。

（4） 安全的和幂等的。所谓安全的意味着该操作用于获取信息而非修改信息。幂等的意味着对同一 URL 的多个请求应该返回同样的结果。

## 10.HTTP1.0 和 HTTP1.1 有什么区别

HTTP 协议老的标准是 HTTP/1.0，目前最通用的标准是 HTTP/1.1。

在同一个 tcp 的连接中可以传送多个 HTTP 请求和响应.

多个请求和响应可以重叠，多个请求和响应可以同时进行.

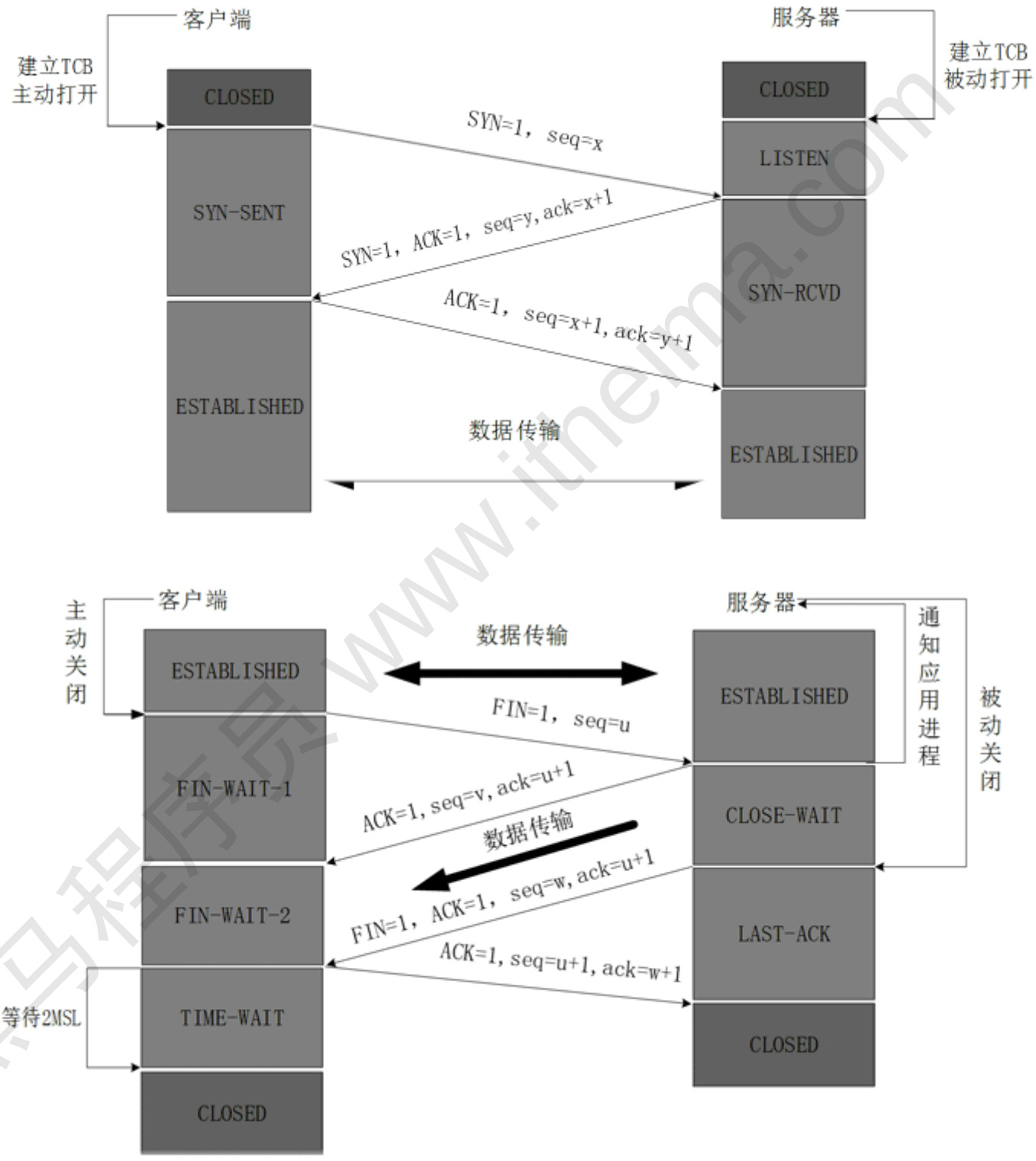
更加多的请求头和响应头(比如 HTTP1.0 没有 host 的字段).

它们最大的区别：

在 HTTP/1.0 中,大多实现为每个请求/响应交换使用新的连接。HTTP 1.0 规定浏览器与服务器只保持短暂的连接，浏览器的每次请求都需要与服务器建立一个 TCP 连接，服务器完成请求处理后立即断开 TCP 连接，服务器不跟踪每个客户也不记录过去的请求。

在 HTTP/1.1 中,一个连接可用于一次或多次请求/响应交换,尽管连接可能由于各种原因被关闭。HTTP 1.1 支持持久连接,在一个 TCP 连接上可以传送多个 HTTP 请求和响应,减少了建立和关闭连接的消耗和延迟。一个包含有许多图像的网页文件的多个请求和应答可以在一个连接中传输，但每个单独的网页文件的请求和应答仍然需要使用各自的连接。HTTP 1.1 还允许客户端不用等待上一次请求结果返回，就可以发出下一次请求，但服务器端必须按照接收到客户端请求的先后顺序依次回送响应结果，以保证客户端能够区分出每次请求的响应内容，这样也显著地减少了整个下载过程所需要的时间。

## 11.TCP 的连接建立过程，以及断开过程？




**12.** 常用协议端口号 **SSH、DHCP、HTTP、FTP、SMTP、DNS** 等？

**13.** 客户端使用 **DHCP** 获取 **IP** 的过程？

发现阶段：即 DHCP 客户端寻找 DHCP 服务器的阶段。

提供阶段：即 DHCP 服务器提供 IP 地址的阶段。

选择阶段：即 DHCP 客户端选择某台 DHCP 服务器提供的 IP 地址的阶段。

确认阶段：即 DHCP 服务器确认所提供的 IP 地址的阶段。

## 14.写出某个网段的网络地址和广播地址？

算法：

子网掩码与 IP 地址进行位与运算，得处网络地址

网络地址 | (~子网掩码)，得出广播地址 （|：位或运算； ~：按位取反）

例如：

IP 地址 10.145.129.20，掩码 255.255.248.0，网络地址和广播地址怎么计算？

网络地址 10.145.128.0 广播地址 10.145.135.255

解答：

IP 转换成二进制：00001010 10010001 10000001 00010010

掩码转换成二进制：11111111 11111111 11111000 00000000

IP 与掩码相与得网络地址（全 1 为 1，见 0 为 0）：00001010 10010001 10000000 00000000

网络地址转换成十进制为：10,145,128,0

看你的掩码把后 24 位的前 13 为划成了子网，后 11 为划成了主机，故：

广播地址则要把网络地址的主机位全换成 1,得：00001010,10010001,10000111,1111111

广播地址转换成十进制为：10,145,135,255

首先由 ip 地址结合子网掩码可以看出的是这是一个由 A 类地址，“借用”13 位的主机位而得到的子网，所以很轻易地得到 网络地址是：10.145.128.0，也即：00001010.10010001.10000 000.00000000（前 21（8+13）位是网络位，后 11 位是主机位）。至于广播地址，网络位+全为 1 的主机位，即得：00001010.10010001.10000 111.11111111，10 进制表达方式就是 10.145.135.255

## 15.什么是 VPN 都有什么类型？

VPN 的隧道协议主要有三种，PPTP、L2TP 和 IPSec，其中 PPTP 和 L2TP 协议工作在 OSI 模型的第二层，又称为二层隧道协议；IPSec 是第三层隧道协议。

## 16.B/S 和 C/S 的区别

b/s 代表浏览器和服务器架构；c/s 代表客户端和服务器架构

网络环境不同（c/s 建立在专用的局域网上，b/s 建立在广域网上）

安全要求不同（c/s 必须安装客户端，安全度较高；b/s 安全度较低）

系统维护不同（c/s 升级困难，需要重新安装最新客户端；b/s 无缝升级）

对系统要求不同（c/s 对系统要求较高；b/s 对系统要求较低）

## 17.线程和进程的区别

进程——资源分配的最小单位，线程——程序执行的最小单位。

线程进程的区别体现在几个方面：

第一：因为进程拥有独立的堆栈空间和数据段，所以每当启动一个新的进程必须分配给它独立的地址空间，建立众多的数据表来维护它的代码段、堆栈段和数据段，这对于多进程来说十分“奢侈”，系统开销比较大，而线程不一样，线程拥有独立的堆栈空间，但是共享数据段，它们彼此之间使用相同的地址空间，共享大部分数据，比进程更节俭，开销比较小，切换速度也比进程快，效率高，但是正由于进程之间独立的特点，使得进程安全性比较高，也因为进程有独立的地址空间，一个进程崩溃后，在保护模式下不会对其它进程产生影响，而线程只是一个进程中的不同执行路径。一个线程死掉就等于整个进程死掉。

第二：体现在通信机制上面，正因为进程之间互不干扰，相互独立，进程的通信机制相对很复杂，譬如管道，信号，消息队列，共享内存，套接字等通信机制，而线程由于共享数据段所以通信机制很方便。。

3.属于同一个进程的所有线程共享该进程的所有资源，包括文件描述符。而不同过的进程相互独立。

4.线程又称为轻量级进程，进程有进程控制块，线程有线程控制块；

5.线程必定也只能属于一个进程，而进程可以拥有多个线程而且至少拥有一个线程；

第四：体现在程序结构上，举一个简明易懂的列子：当我们使用进程的时候，我们不自主的使用 if else 嵌套来判断 pid，使得程序结构繁琐，但是当我们使用线程的时候，基本上可以甩掉它，当然程序内部执行功能单元需要使用的时候还是要使用，所以线程对程序结构的改善有很大帮助。

## 18.常用的响应码

HTTP 响应码，也称 http 状态码(HTTP Status Code)，反映了 web 服务器处理 HTTP 请求状态，每一个响应码都代表了一种服务端反馈的响应状态，标识了本次请求是否成功。我们应该了解常见的响应码代表的状态，通过响应码能够对错误进行排查和定位，这是一个测试的必备技能~HTTP 响应码通常分为五大类：

1XX——信息类（Information），表示收到 http 请求，正在进行下一步处理，通常是一种瞬间的响应状态

2XX——成功类（Successful），表示用户请求被正确接收、理解和处理

200（OK）：请求成功。一般用于 GET 与 POST 请求

201（Created）：已创建。成功请求并创建了新的资源

202（Accepted）：

3XX——重定向类（Redirection），表示没有请求成功，必须采取进一步的动作

301（Moved Permanently）：资源被永久移动。请求的资源已被永久的移动到新 URI，返回信息会包括新的 URI，浏览器会自动定向到新 URI。今后任何新的请求都应使用新的 URI

302（Found）：资源临时移动。资源只是临时被移动，客户端应继续使用原有 URI

304：用其他策略获取资源

4XX——客户端错误（Client Error），表示客户端提交的请求包含语法错误或不能正确执行

400（Bad Requests）：客户端请求的地址不存在或者包含不支持的参数

401（Unauthorized）：未授权，或认证失败。对于需要登录的网页，服务器可能返回此响应

403（Forbidden）：没权限。服务器收到请求，但拒绝提供服务

404（Not Found）：请求的资源不存在。遇到 404 首先检查请求 url 是否正确

5XX——服务端错误（Server Error），表示服务器不能正确执行一个正确的请求（客户端请求的方法及参数是正确的，服务端不能正确执行，如网络超时、服务僵死，可以查看服务端日志再进一步解决）

500（Internal Server Error）：服务器内部错误，无法完成请求

503（Service Unavailable）：由于超载或系统维护（一般是访问人数过多），服务器无法处理客户端的请求 ，通常这只是暂时状态

# 三、 组成原理

## 1. 计算机基本组成

A. 冯·诺依曼计算机的特点（机器以运算器为中心）

1. 计算机由运算器、存储器、控制器、输入设备和输出设备五大部件组成

2. 指令(程序)和数据以二进制不加区别地存储在存储器中

3. 程序自动运行

B.现代计算机由三大部分组成（已经转化为以存储器为中心）

1. CPU(Central Processing Unit) 中央处理器，核心部件为 ALU(Arithmetic Logic Unit,算术逻辑单元)和 CU(Control Unit,控制单元)

2. I/O 设备(受 CU 控制)

3. 主存储器(Main Memory,MM)，分为 RAM(随机存储器)和 ROM(只读存储器)

## 2.一条指令在 CPU 的执行过程

1. Ad(Address) 形式地址

2. DR(Data Register) 数据寄存器

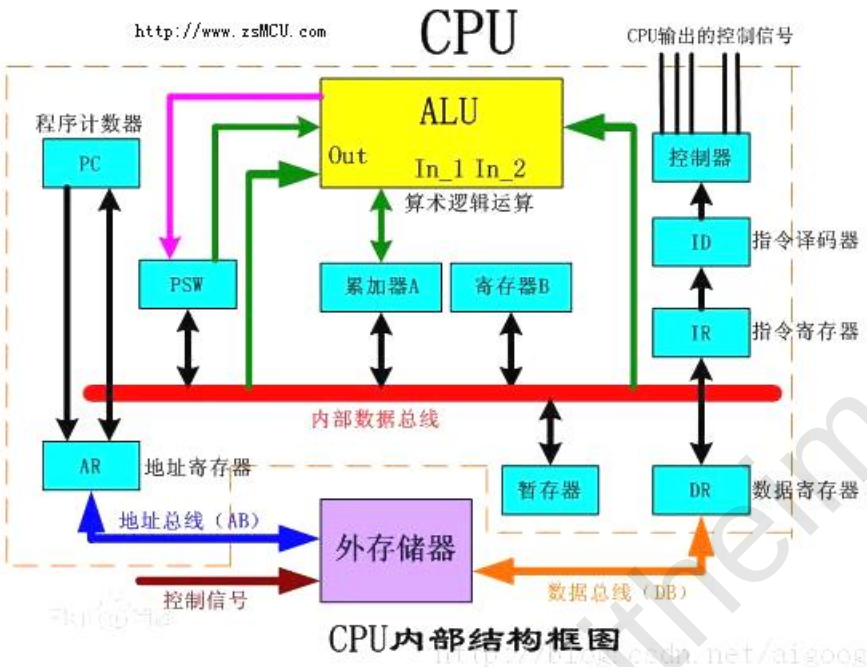
3. AR(Address Register) 地址寄存器(MAR)

4. IR(Instruction Register) 指令寄存器

5. BR(Buffer Register) 缓冲寄存器(MBR)

5. ID(Instruction Decoder) 指令译码器

6. PC(ProgramCounter) 程序计数器




CPU内部结构框图

过程详述：

几乎所有的冯·诺伊曼型计算机的 CPU，其工作都可以分为 5 个阶段：

取指令

指令译码

执行指令

访存取数

结果写回

1.取指令阶段

取指令（Instruction Fetch，IF）阶段是将一条指令从主存中取到指令寄存器的过程。

程序计数器 PC 中的数值，用来指示当前指令在主存中的位置。当一条指令被取出后，PC 中的数值将根据指令字长度而自动递增：若为单字长指令，则(PC)+1àPC；若为双字长指令，则(PC)+2àPC，依此类推。

//PC -> AR -> Memory

//Memory -> IR

2.指令译码阶段

取出指令后，计算机立即进入指令译码（Instruction Decode，ID）阶段。

在指令译码阶段，指令译码器按照预定的指令格式，对取回的指令进行拆分和解释，识别区分出不

同的指令类别以及各种获取操作数的方法。

在组合逻辑控制的计算机中，指令译码器对不同的指令操作码产生不同的控制电位，以形成不同的微操作序列；在微程序控制的计算机中，指令译码器用指令操作码来找到执行该指令的微程序的入口，并从此入口开始执行。

//　　{1.Ad

//Memory -> IR -> ID ->　　{2.PC 变化

//　　{3.CU（Control Unit）

3.访存取数阶段

根据指令需要，有可能要访问主存，读取操作数，这样就进入了访存取数（Memory，MEM）阶段。

此阶段的任务是：根据指令地址码，得到操作数在主存中的地址，并从主存中读取该操作数用于运算。

//Ad -> AR -> AD -> Memory

4.执行指令阶段

在取指令和指令译码阶段之后，接着进入执行指令（Execute，EX）阶段。

此阶段的任务是完成指令所规定的各种操作，具体实现指令的功能。为此，CPU 的不同部分被连接起来，以执行所需的操作。

例如，如果要求完成一个加法运算，算术逻辑单元 ALU 将被连接到一组输入和一组输出，输入端提供需要相加的数值，输出端将含有最后的运算结果。

//Memory -> DR -> ALU

5.结果写回阶段

作为最后一个阶段，结果写回（Writeback，WB）阶段把执行指令阶段的运行结果数据“写回”到某种存储形式：结果数据经常被写到 CPU 的内部寄存器中，以便被后续的指令快速地存取；在有些情况下，结果数据也可被写入相对较慢、但较廉价且容量较大的主存。许多指令还会改变程序状态字寄存器中标志位的状态，这些标志位标识着不同的操作结果，可被用来影响程序的动作。

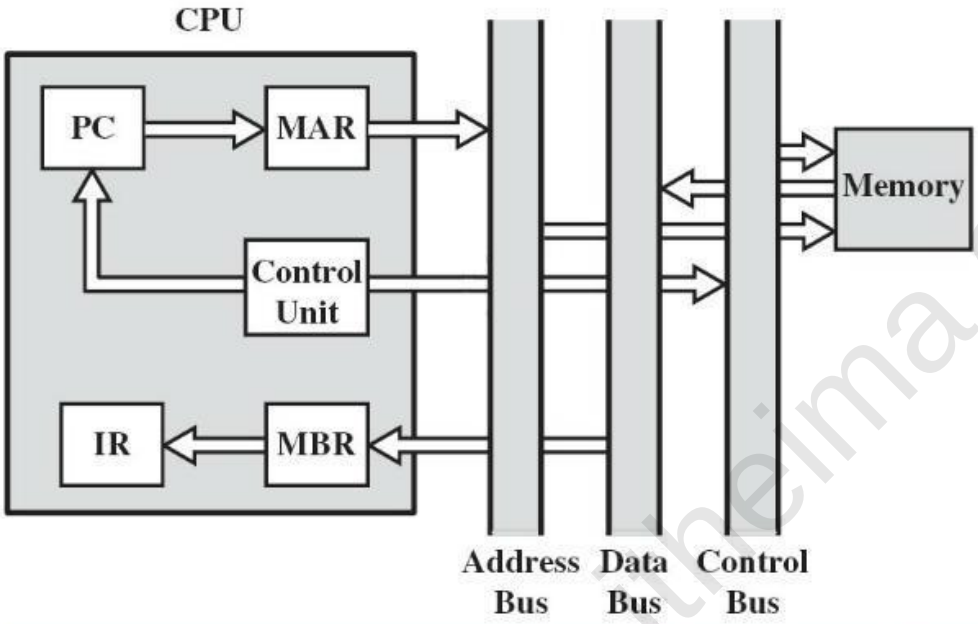
//DR -> Memory

6.循环阶段

在指令执行完毕、结果数据写回之后，若无意外事件（如结果溢出等）发生，计算机就接着从程序计数器 PC 中取得下一条指令地址，开始新一轮的循环，下一个指令周期将顺序取出下一条指令。

//重复 1~5

//遇 hlt(holt on)停止




指令的执行过程



### 3.计算机的逻辑部件

1. 若逻辑电路的输出状态仅和当时的输入状态有关，而与过去的输入状态无关，则称这种逻辑电路为组合逻辑电路

常见的组合逻辑电路有三态门、异或门、加法器、算术逻辑单元、译码器

2. 若逻辑电路的输出状态不但与当时的输入状态有关，而且还与电路在此前的输入状态有关，则称这种逻辑电路为时序逻辑电路。时序电路内必须有能存储信息的记忆元件即触发器。触发器是构成时序电路的基础。

3. “阵列”是逻辑元件在硅芯片上以阵列形式排列。常见阵列逻辑电路有:只读存储器 ROM、可编程序逻辑阵列 PLA、可编程序阵列逻辑 PAL、通用阵列逻辑 GAL 等

## 四、 数据结构与算法

### 1. 冒泡排序

冒泡思想：通过无序区中相邻记录的关键字间的比较和位置的交换，使关键字最小的记录像气泡一样逐渐向上漂至水面。整个算法是从最下面的记录开始，对每两个相邻的关键字进行比较，把关键字较小的记录

放到关键字较大的记录的上面，经过一趟排序后，关键字最小的记录到达最上面，接着再在剩下的记录中找关键字次小的记录，把它放在第二个位置上，依次类推，一直到所有记录有序为止

```
def bsort(alist):
    l = len(alist)
    for i in range(l-1):
        flag = 1
        for j in range(l-i-1):
            if alist[j] > alist[j+1]:
                alist[j],alist[j+1] = (alist[j+1],alist[j])
                flag = 0
        if flag == 1:
            return alist
```

## 2. 插入排序

插入排序的基本操作就是将一个数据插入到已经排好序的有序数据中，从而得到一个新的、个数加一的有序数据，算法适用于少量数据的排序，时间复杂度为 O(n^2)。是稳定的排序方法。插入算法把要排序的数组分成两部分：第一部分包含了这个数组的所有元素，但将最后一个元素除外（让数组多一个空间才有插入的位置），而第二部分就只包含这一个元素（即待插入元素）。在第一部分排序完成后，再将这个最后元素插入到已排好序的第一部分中。

```
def insert_sort(lists):
    # 插入排序
    count = len(lists)
    for i in range(1, count):
        key = lists[i]
        j = i - 1
        while j >= 0:
            if lists[j] > key:
                lists[j + 1] = lists[j]
                lists[j] = key
            j -= 1
    return lists
```

## 3. 希尔排序

希尔排序(Shell Sort)是插入排序的一种。也称缩小增量排序，是直接插入排序算法的一种更高效的改进版本。希尔排序是非稳定排序算法。该方法因 DL．Shell 于 1959 年提出而得名。 希尔排序是把记录按下标的一定增量分组，对每组使用直接插入排序算法排序；随着增量逐渐减少，每组包含的关键词越来越多，当增量减至 1 时，整个文件恰被分成一组，算法便终止。

```
def shell_sort(lists):
    # 希尔排序
    count = len(lists)
    step = 2
    group = count / step
    while group > 0:
        for i in range(0, group):
            j = i + group
            while j < count:
                k = j - group
                key = lists[j]
                while k >= 0:
                    if lists[k] > key:
                        lists[k + group] = lists[k]
                        lists[k] = key
                    k -= group
                j += group
        group /= step
    return lists
```

## 4. 快速排序

通过一趟排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据都要小，然后再按此方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行，以此达到整个数据变成有序序列。

```
def quick_sort(lists, left, right):
    # 快速排序
    if left >= right:
        return lists
    key = lists[left]
    low = left
    high = right
    while left < right:
        while left < right and lists[right] >= key:
            right -= 1
        lists[left] = lists[right]
        while left < right and lists[left] <= key:
            left += 1
        lists[right] = lists[left]
    lists[right] = key
    quick_sort(lists, low, left - 1)
    quick_sort(lists, left + 1, high)
    return lists
```

## 5. 直接选择排序

基本思想：第 1 趟，在待排序记录 r1 ~ r[n]中选出最小的记录，将它与 r1 交换；第 2 趟，在待排序记录 r2 ~ r[n]中选出最小的记录，将它与 r2 交换；以此类推，第 i 趟在待排序记录 r[i] ~ r[n]中选出最小的记录，将它与 r[i]交换，使有序序列不断增长直到全部排序完毕。

```
def select_sort(lists):
    # 选择排序
```

```
    count = len(lists)
    for i in range(0, count):
        min = i
        for j in range(i + 1, count):
            if lists[min] > lists[j]:
                min = j
        lists[min], lists[i] = lists[i], lists[min]
    return lists
```

## 6. 堆排序

堆排序(Heapsort)是指利用堆积树（堆）这种数据结构所设计的一种排序算法，它是选择排序的一种。可以利用数组的特点快速定位指定索引的元素。堆分为大根堆和小根堆，是完全二叉树。大根堆的要求是每个节点的值都不大于其父节点的值，即 A[PARENT[i]] >= A[i]。在数组的非降序排序中，需要使用的就是大根堆，因为根据大根堆的要求可知，最大的值一定在堆顶。

```
2   def adjust_heap(lists, i, size):
3       lchild = 2 * i + 1
4       rchild = 2 * i + 2
5       max = i
6       if i < size / 2:
7           if lchild < size and lists[lchild] > lists[max]:
8               max = lchild
9           if rchild < size and lists[rchild] > lists[max]:
10              max = rchild
11          if max != i:
12              lists[max], lists[i] = lists[i], lists[max]
13              adjust_heap(lists, max, size)
14
15  def build_heap(lists, size):
16      for i in range(0, (size/2))[::-1]:
17          adjust_heap(lists, i, size)
18
19  def heap_sort(lists):
20      size = len(lists)
21      build_heap(lists, size)
22      for i in range(0, size)[::-1]:
```

```
23          lists[0], lists[i] = lists[i], lists[0]
            adjust_heap(lists, 0, i)
```

## 7. 归并排序

归并排序是建立在归并操作上的一种有效的排序算法,该算法是采用分治法（Divide and Conquer）的一个非常典型的应用。将已有序的子序列合并，得到完全有序的序列；即先使每个子序列有序，再使子序列段间有序。若将两个有序表合并成一个有序表，称为二路归并。

归并过程为：比较 a[i]和 a[j]的大小，若 a[i]≤a[j]，则将第一个有序表中的元素 a[i]复制到 r[k]中，并令 i 和 k 分别加上 1；否则将第二个有序表中的元素 a[j]复制到 r[k]中，并令 j 和 k 分别加上 1，如此循环下去，直到其中一个有序表取完，然后再将另一个有序表中剩余的元素复制到 r 中从下标 k 到下标 t 的单元。归并排序的算法我们通常用递归实现，先把待排序区间[s,t]以中点二分，接着把左边子区间排序，再把右边子区间排序，最后把左区间和右区间用一次归并操作合并成有序的区间[s,t]。

```
def merge(left, right):
    i, j = 0, 0
    result = []
    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1
    result += left[i:]
    result += right[j:]
    return result


def merge_sort(lists):
    # 归并排序
    if len(lists) <= 1:
```

```
        return lists
    num = len(lists) / 2
    left = merge_sort(lists[:num])
    right = merge_sort(lists[num:])
    return merge(left, right)
```

## 8. 基数排序

基数排序（radix sort）属于“分配式排序”（distribution sort），又称“桶子法”（bucket sort）或 bin sort，顾名思义，它是透过键值的部份资讯，将要排序的元素分配至某些“桶”中，藉以达到排序的作用，基数排序法是属于稳定性的排序，其时间复杂度为 O (nlog(r)m)，其中 r 为所采取的基数，而 m 为堆数，在某些时候，基数排序法的效率高于其它的稳定性排序法。

```
import math
def radix_sort(lists, radix=10):
    k = int(math.ceil(math.log(max(lists), radix)))
    bucket = [[] for i in range(radix)]
    for i in range(1, k+1):
        for j in lists:
            bucket[j/(radix**(i-1)) % (radix**i)].append(j)
        del lists[:]
        for z in bucket:
            lists += z
            del z[:]
    return lists
```